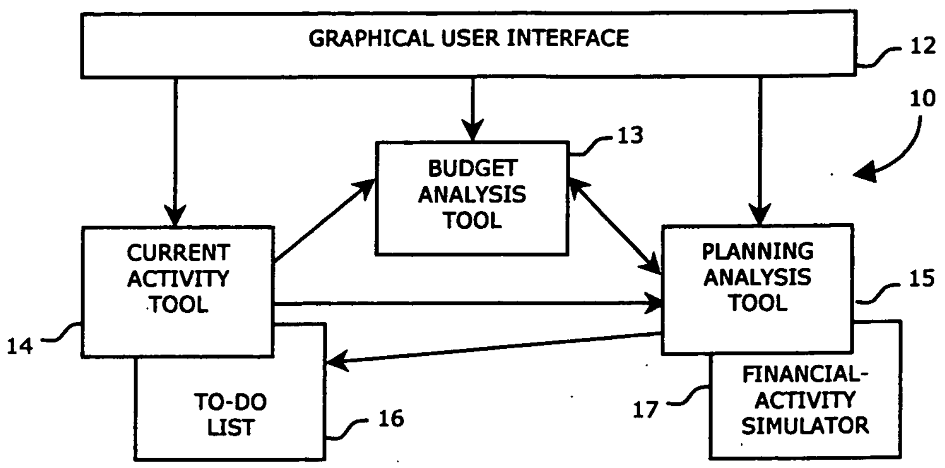

GRAPHICAL USER INTERFACE
12
10
BUDGET ANALYSIS TOOL
13
CURRENT ACTIVITY TOOL
14
PLANNING ANALYSIS TOOL
15
TO-DO LIST
16
FINANCIAL-ACTIVITY SIMULATOR
17

FIG. 1

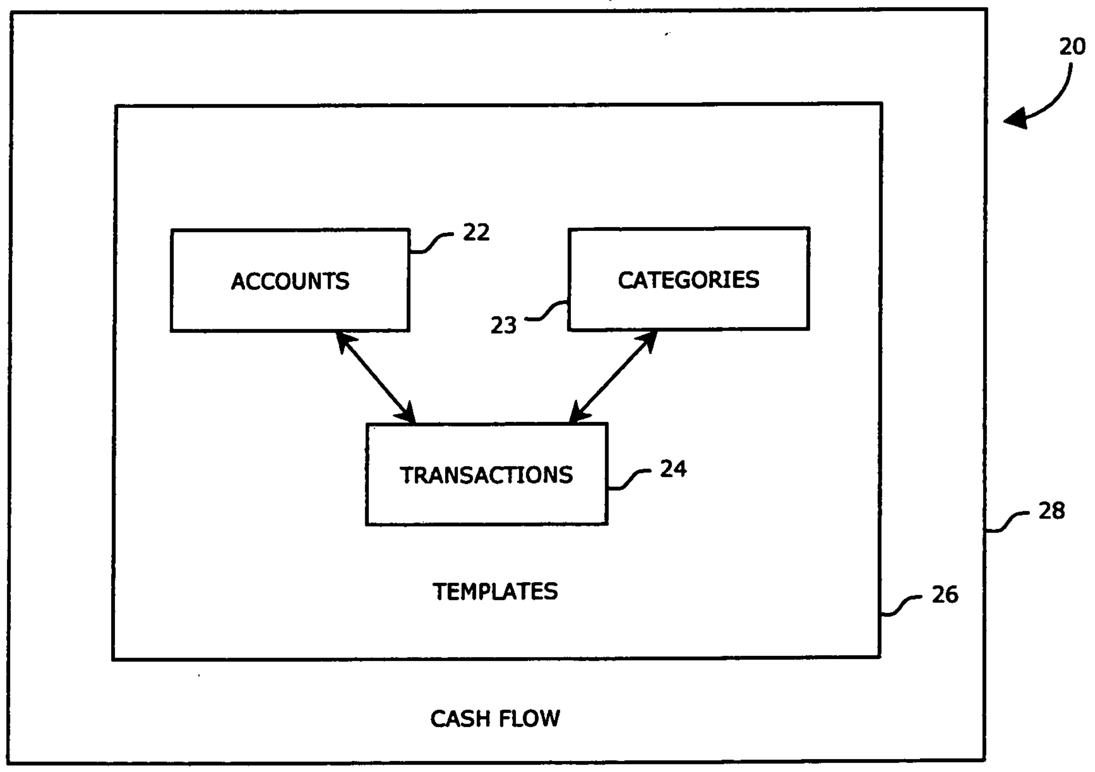

20
ACCOUNTS
22
CATEGORIES
23
TRANSACTIONS
24
TEMPLATES
26
CASH FLOW
28

FIG. 2

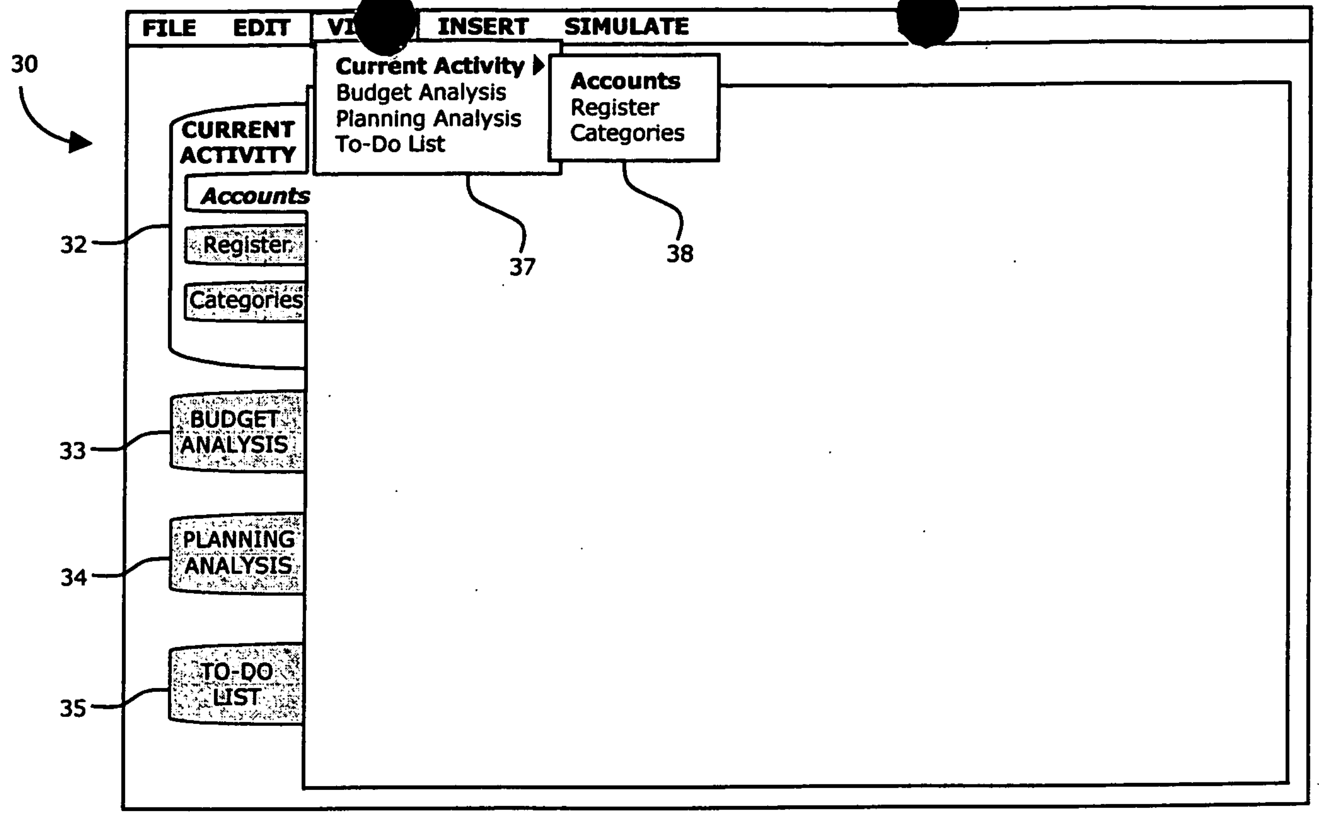
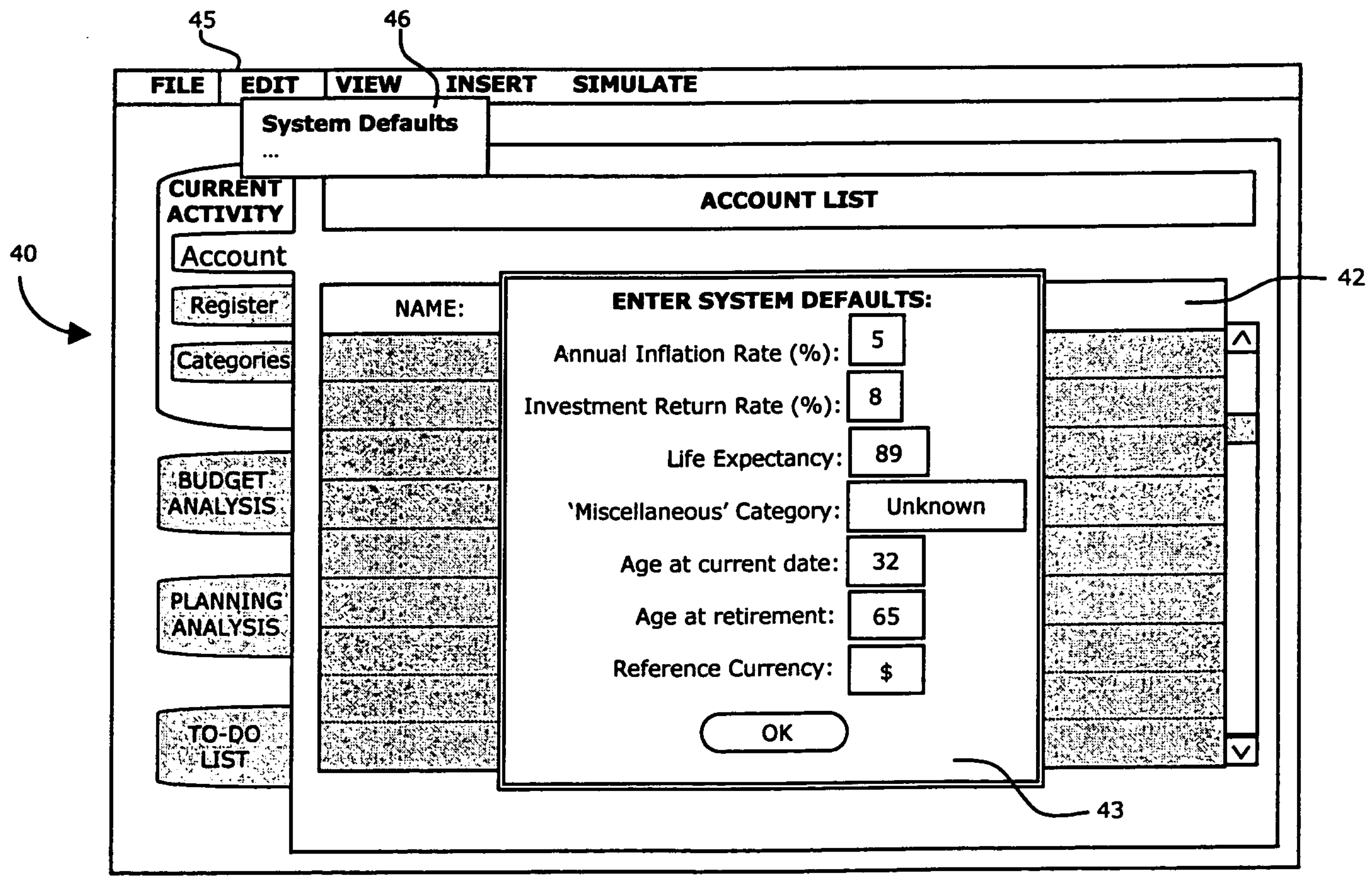


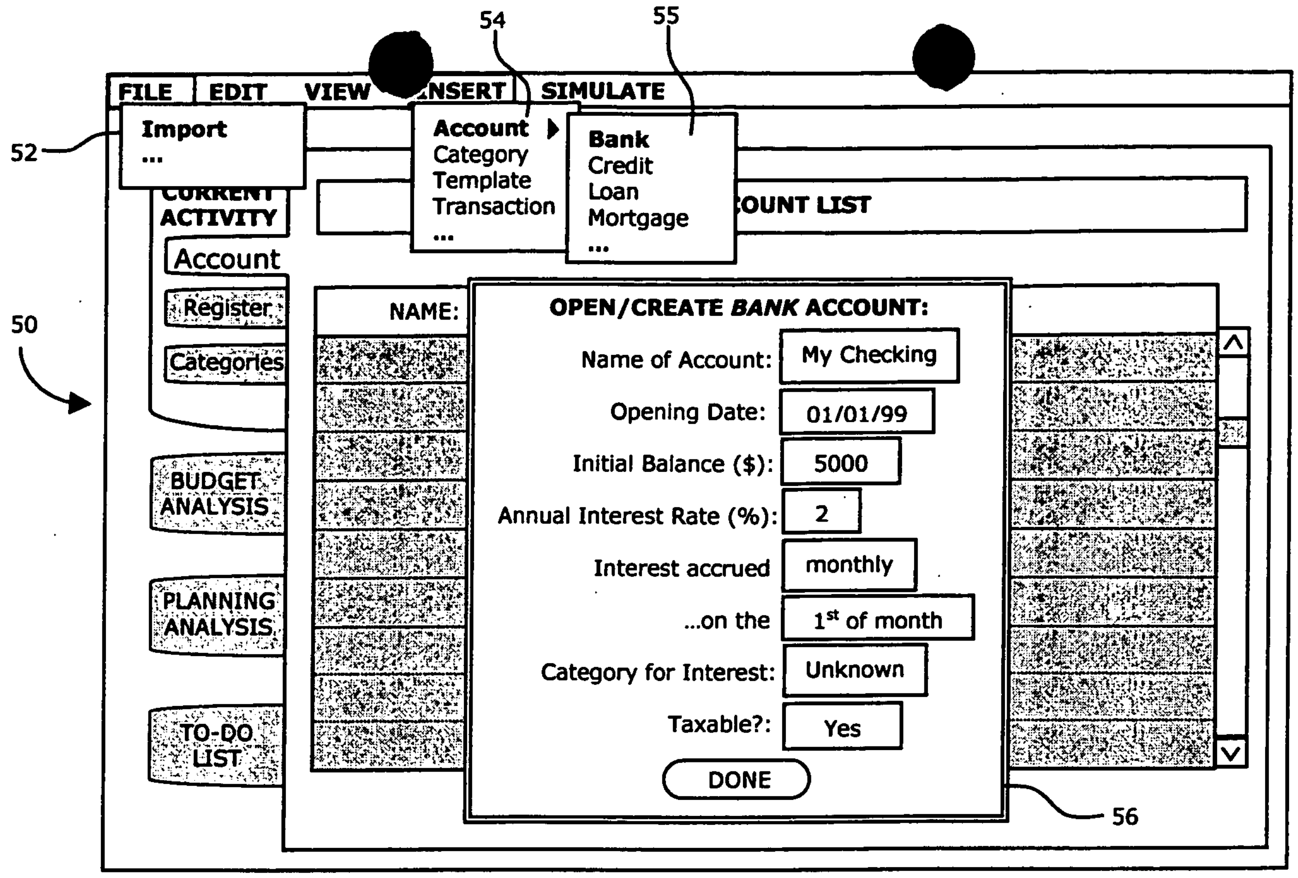
FIG. 3



ENTER SYSTEM DEFAULTS:

Annual Inflation Rate (%):	5
Investment Return Rate (%):	8
Life Expectancy:	89
'Miscellaneous' Category:	Unknown
Age at current date:	32
Age at retirement:	65
Reference Currency:	$

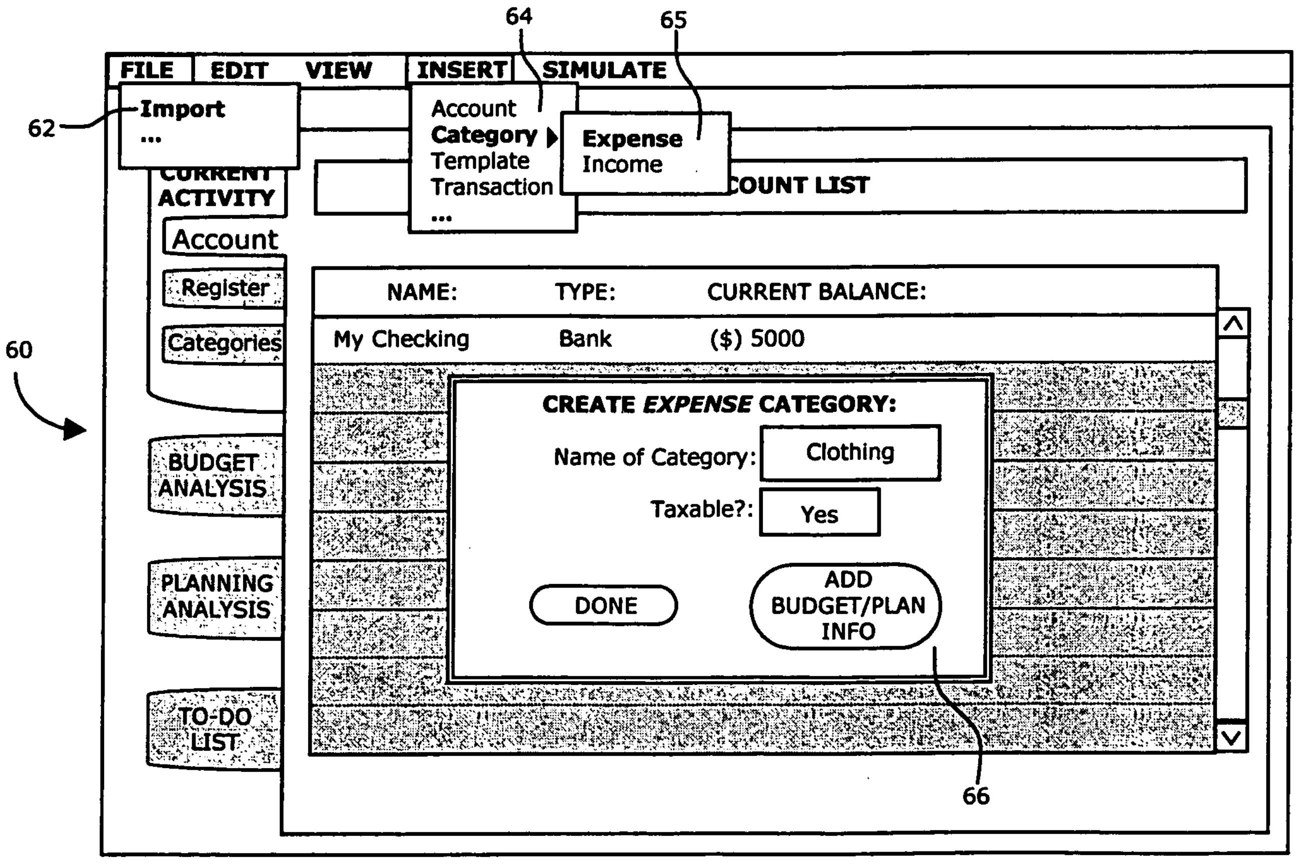
FIG. 4



OPEN/CREATE *BANK* ACCOUNT:

Name of Account:	My Checking
Opening Date:	01/01/99
Initial Balance ($):	5000
Annual Interest Rate (%):	2
Interest accrued	monthly
...on the	1st of month
Category for Interest:	Unknown
Taxable?:	Yes

FIG. 5



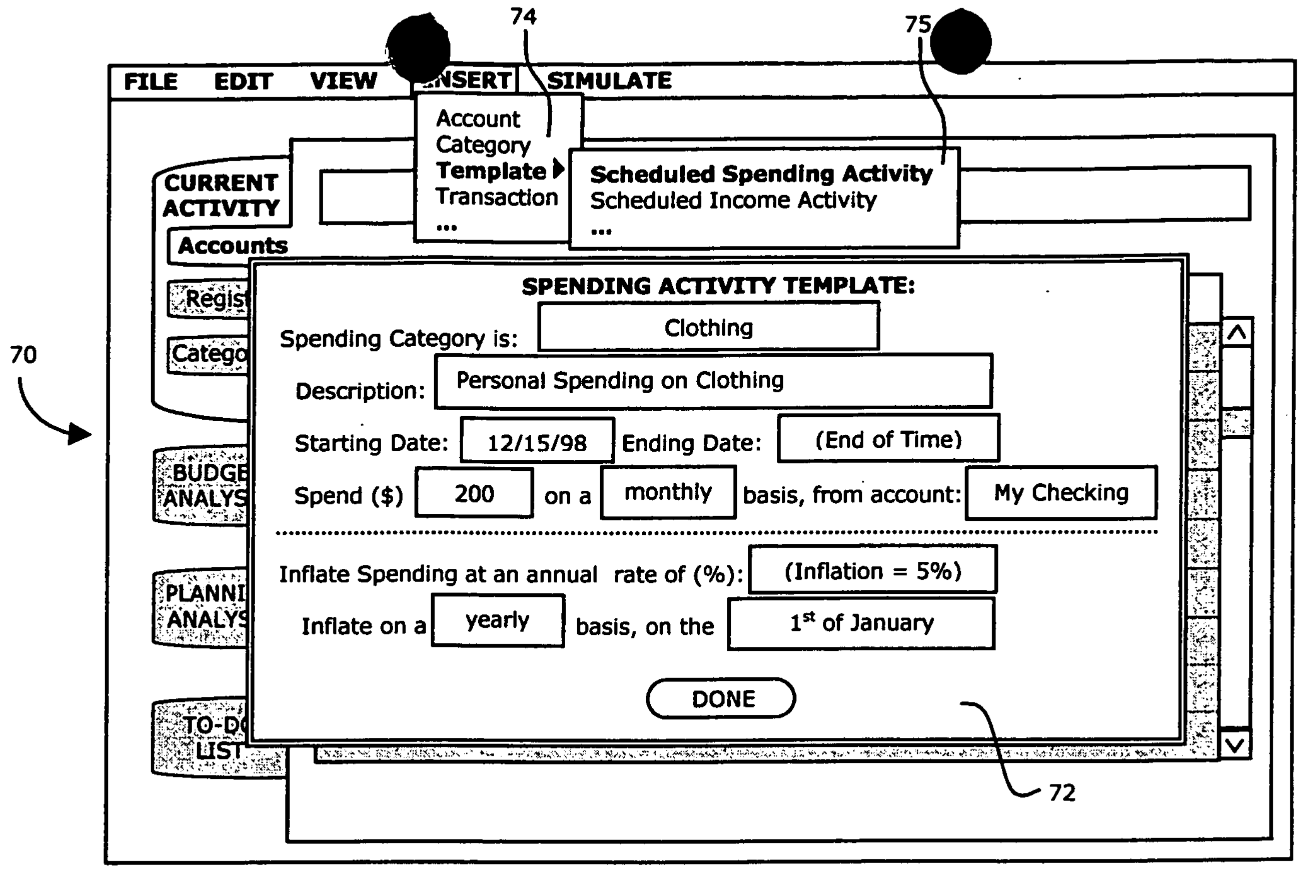
NAME:	TYPE:	CURRENT BALANCE:
My Checking	Bank	($) 5000

CREATE *EXPENSE* CATEGORY:

Name of Category:	Clothing
Taxable?:	Yes

FIG. 6

BEST AVAILABLE COPY



SPENDING ACTIVITY TEMPLATE:

Spending Category is: Clothing

Description: Personal Spending on Clothing

Starting Date: 12/15/98 Ending Date: (End of Time)

Spend ($) 200 on a monthly basis, from account: My Checking

Inflate Spending at an annual rate of (%): (Inflation = 5%)

Inflate on a yearly basis, on the 1st of January

FIG. 7

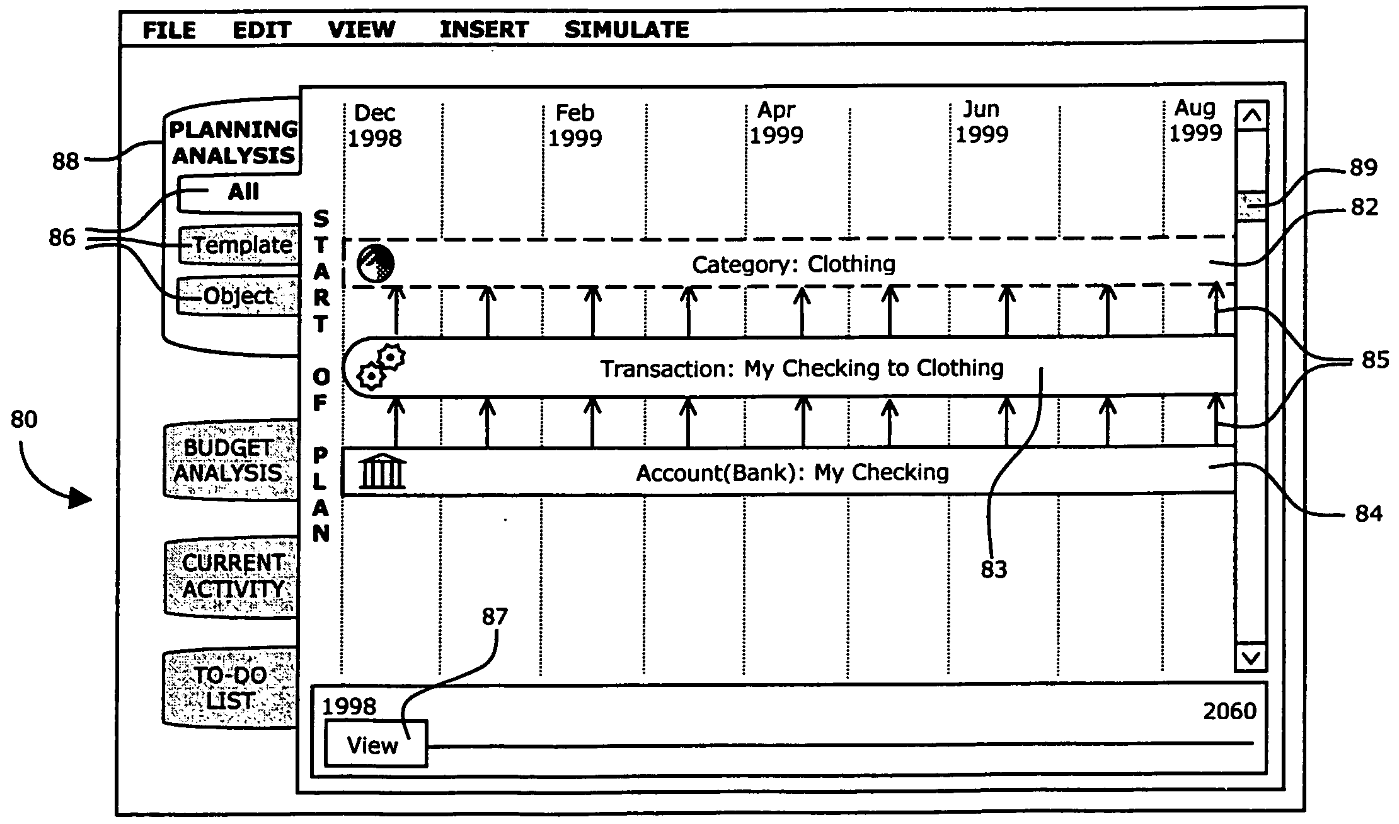


FIG. 8

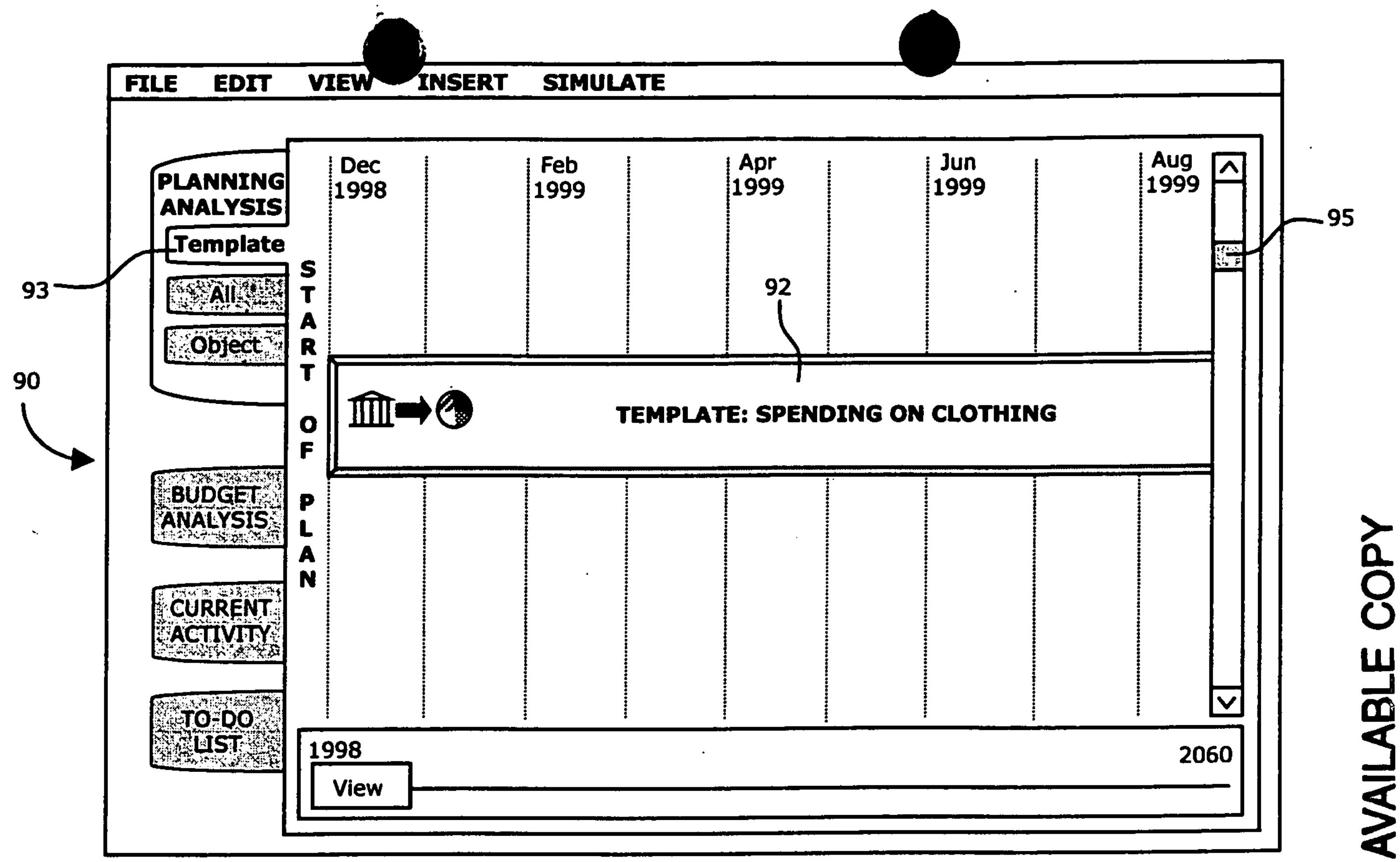
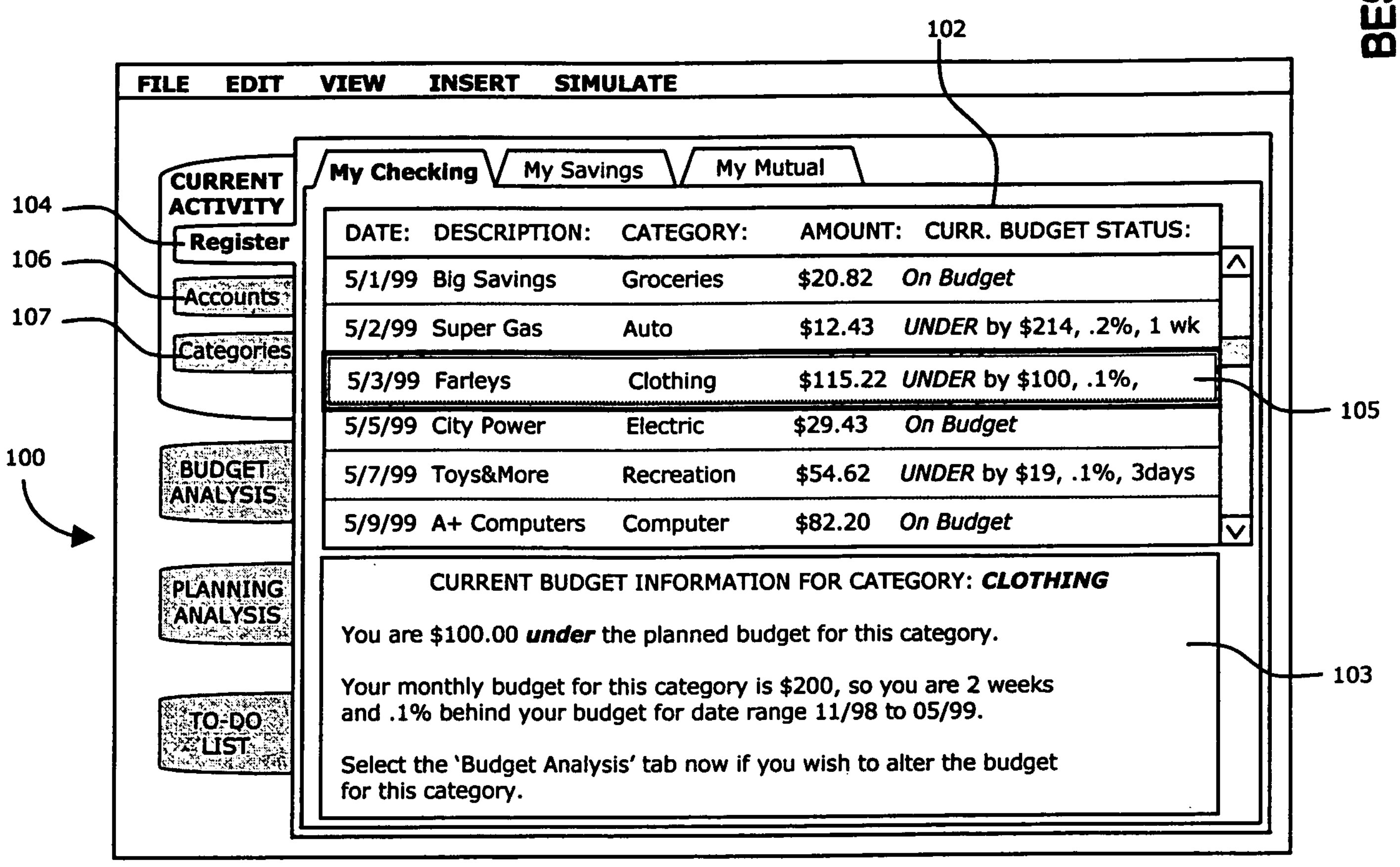


FIG. 9



DATE:	DESCRIPTION:	CATEGORY:	AMOUNT:	CURR. BUDGET STATUS:
5/1/99	Big Savings	Groceries	$20.82	*On Budget*
5/2/99	Super Gas	Auto	$12.43	*UNDER* by $214, .2%, 1 wk
5/3/99	Farleys	Clothing	$115.22	*UNDER* by $100, .1%,
5/5/99	City Power	Electric	$29.43	*On Budget*
5/7/99	Toys&More	Recreation	$54.62	*UNDER* by $19, .1%, 3days
5/9/99	A+ Computers	Computer	$82.20	*On Budget*

CURRENT BUDGET INFORMATION FOR CATEGORY: ***CLOTHING***

You are $100.00 ***under*** the planned budget for this category.

Your monthly budget for this category is $200, so you are 2 weeks and .1% behind your budget for date range 11/98 to 05/99.

Select the 'Budget Analysis' tab now if you wish to alter the budget for this category.

FIG. 10

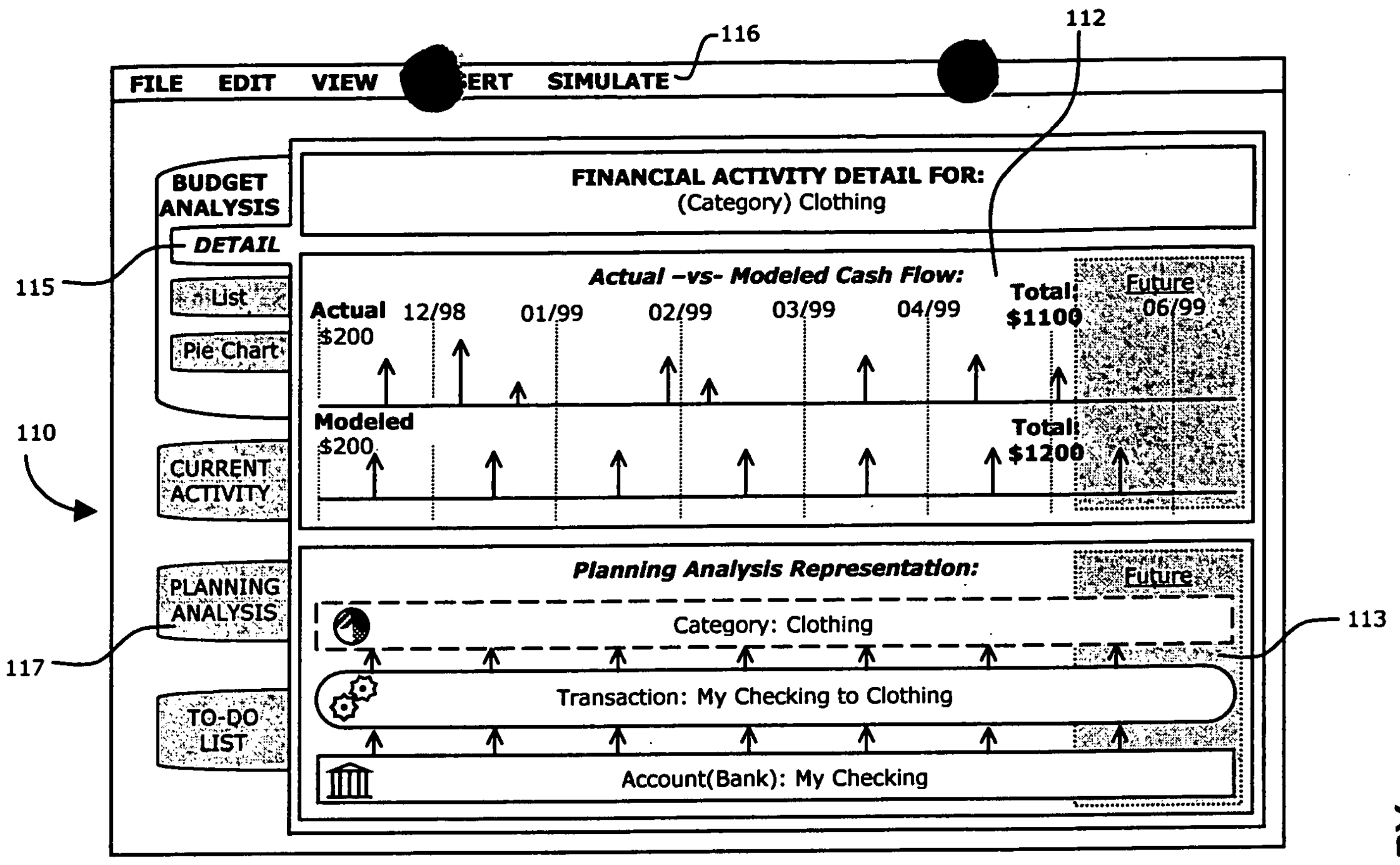
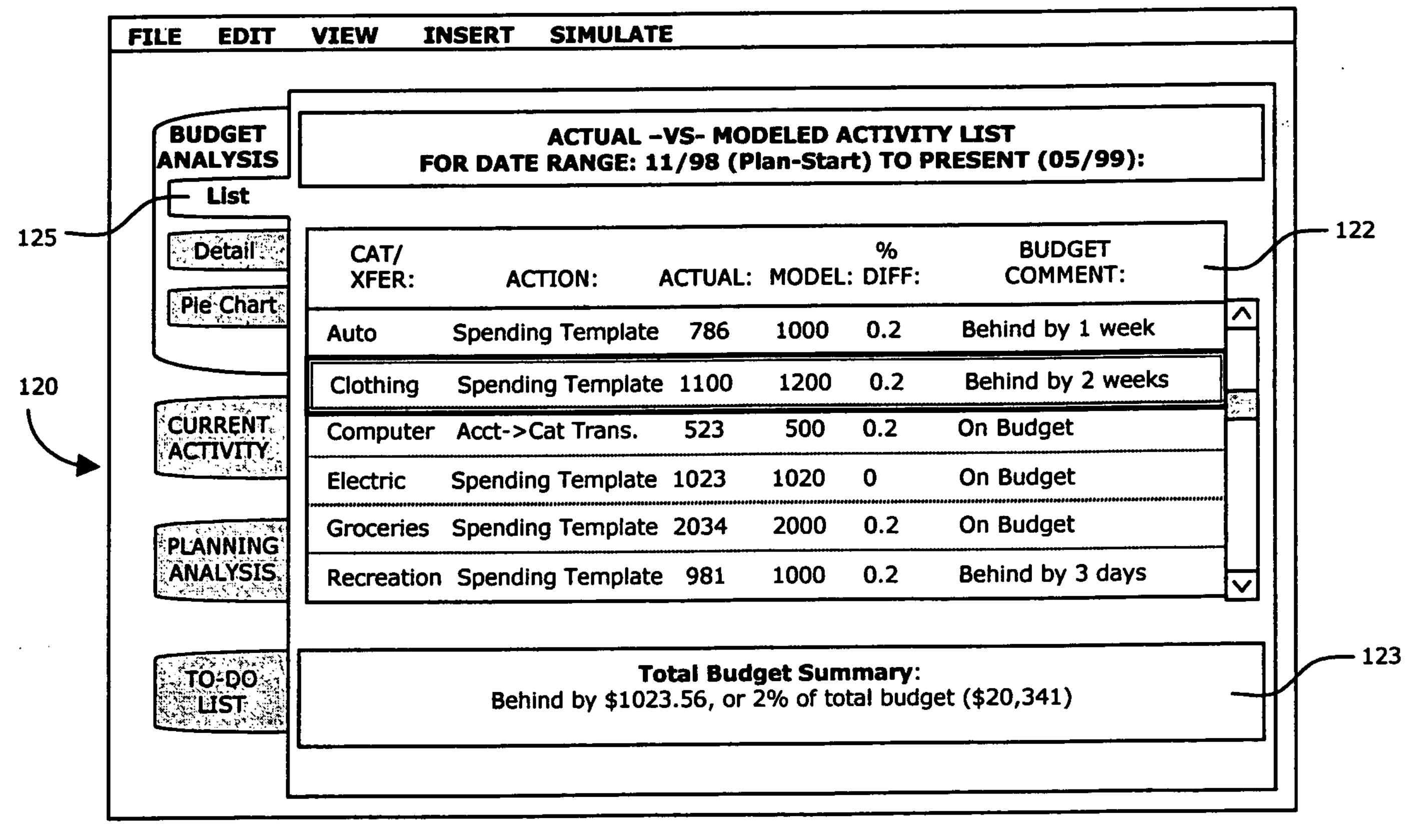


FIG. 11

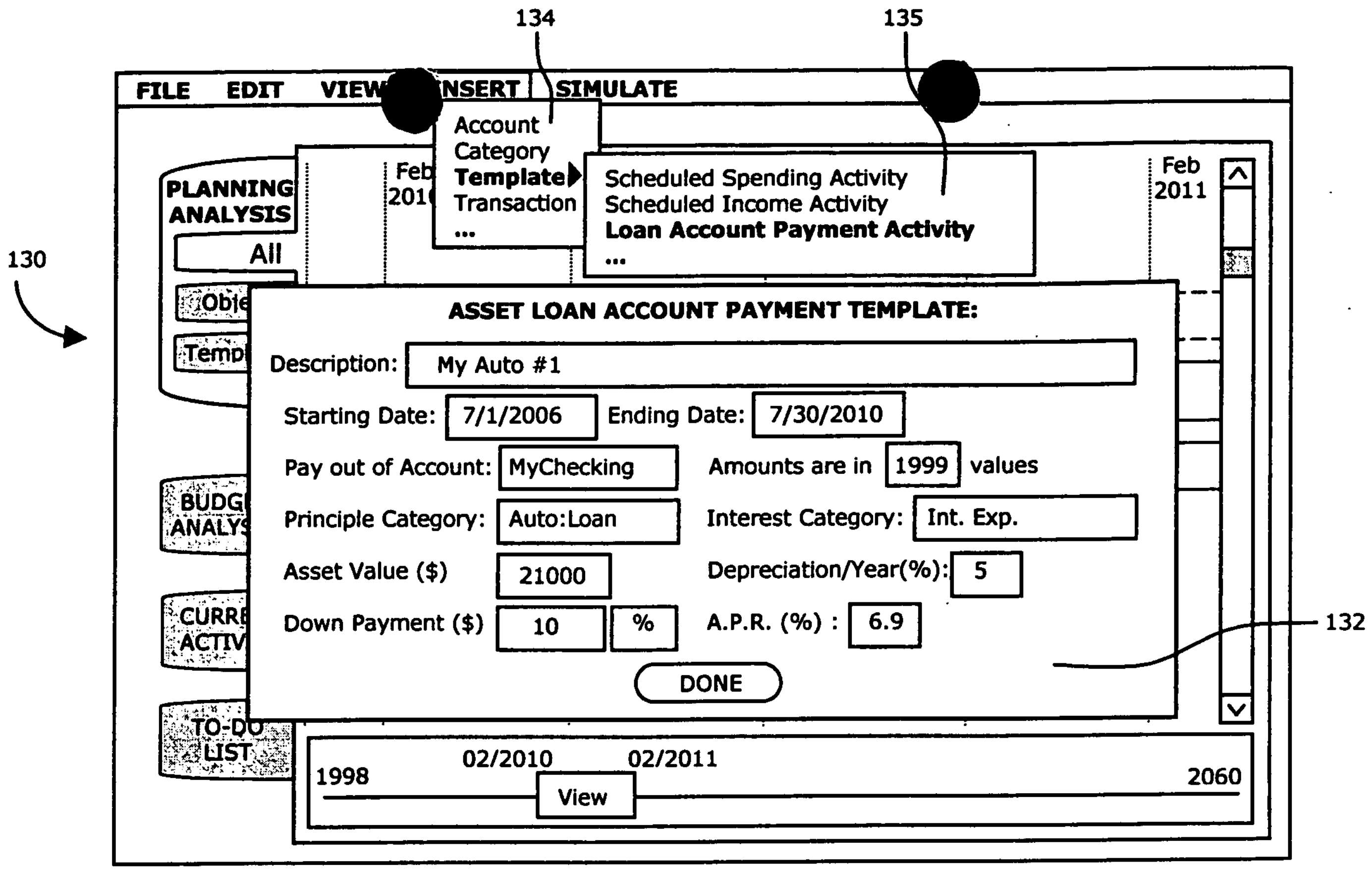


ACTUAL -VS- MODELED ACTIVITY LIST
FOR DATE RANGE: 11/98 (Plan-Start) TO PRESENT (05/99):

CAT/ XFER:	ACTION:	ACTUAL:	MODEL:	% DIFF:	BUDGET COMMENT:
Auto	Spending Template	786	1000	0.2	Behind by 1 week
Clothing	Spending Template	1100	1200	0.2	Behind by 2 weeks
Computer	Acct->Cat Trans.	523	500	0.2	On Budget
Electric	Spending Template	1023	1020	0	On Budget
Groceries	Spending Template	2034	2000	0.2	On Budget
Recreation	Spending Template	981	1000	0.2	Behind by 3 days

Total Budget Summary:
Behind by $1023.56, or 2% of total budget ($20,341)

FIG. 12



ASSET LOAN ACCOUNT PAYMENT TEMPLATE:

Description: My Auto #1
Starting Date: 7/1/2006 Ending Date: 7/30/2010
Pay out of Account: MyChecking Amounts are in 1999 values
Principle Category: Auto:Loan Interest Category: Int. Exp.
Asset Value ($) 21000 Depreciation/Year(%): 5
Down Payment ($) 10 % A.P.R. (%) : 6.9
DONE

FIG. 13

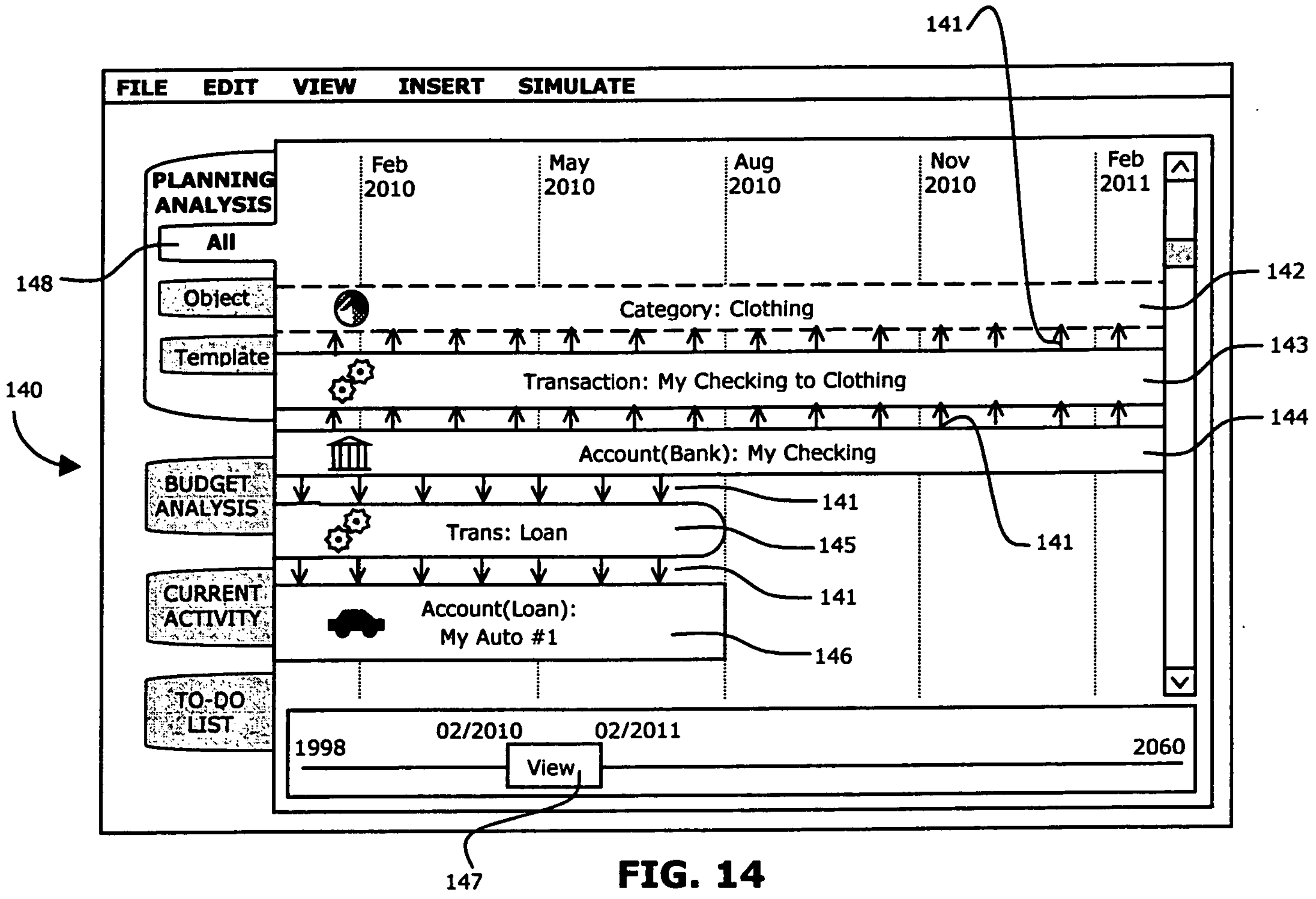


FIG. 14

BEST AVAILABLE COPY

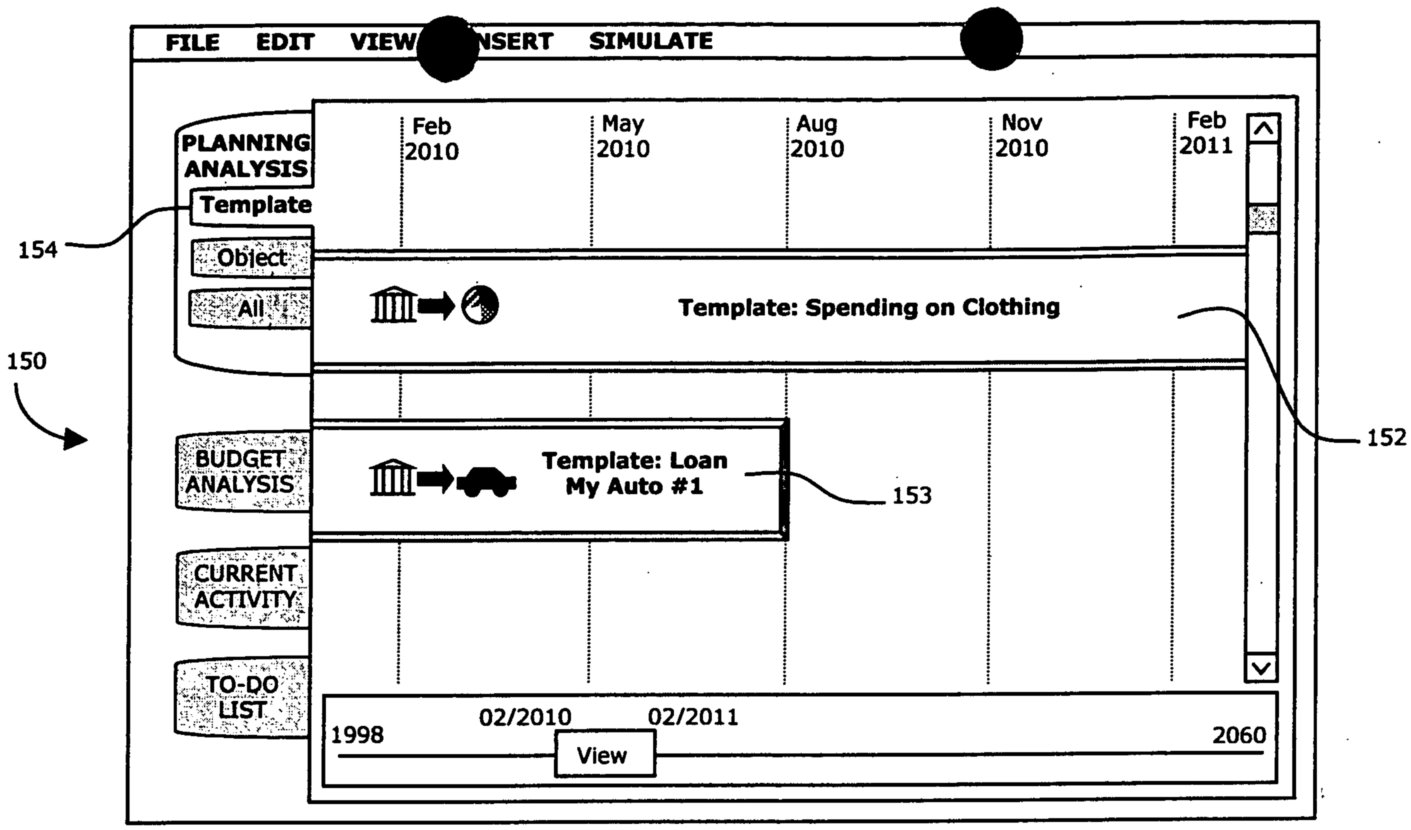
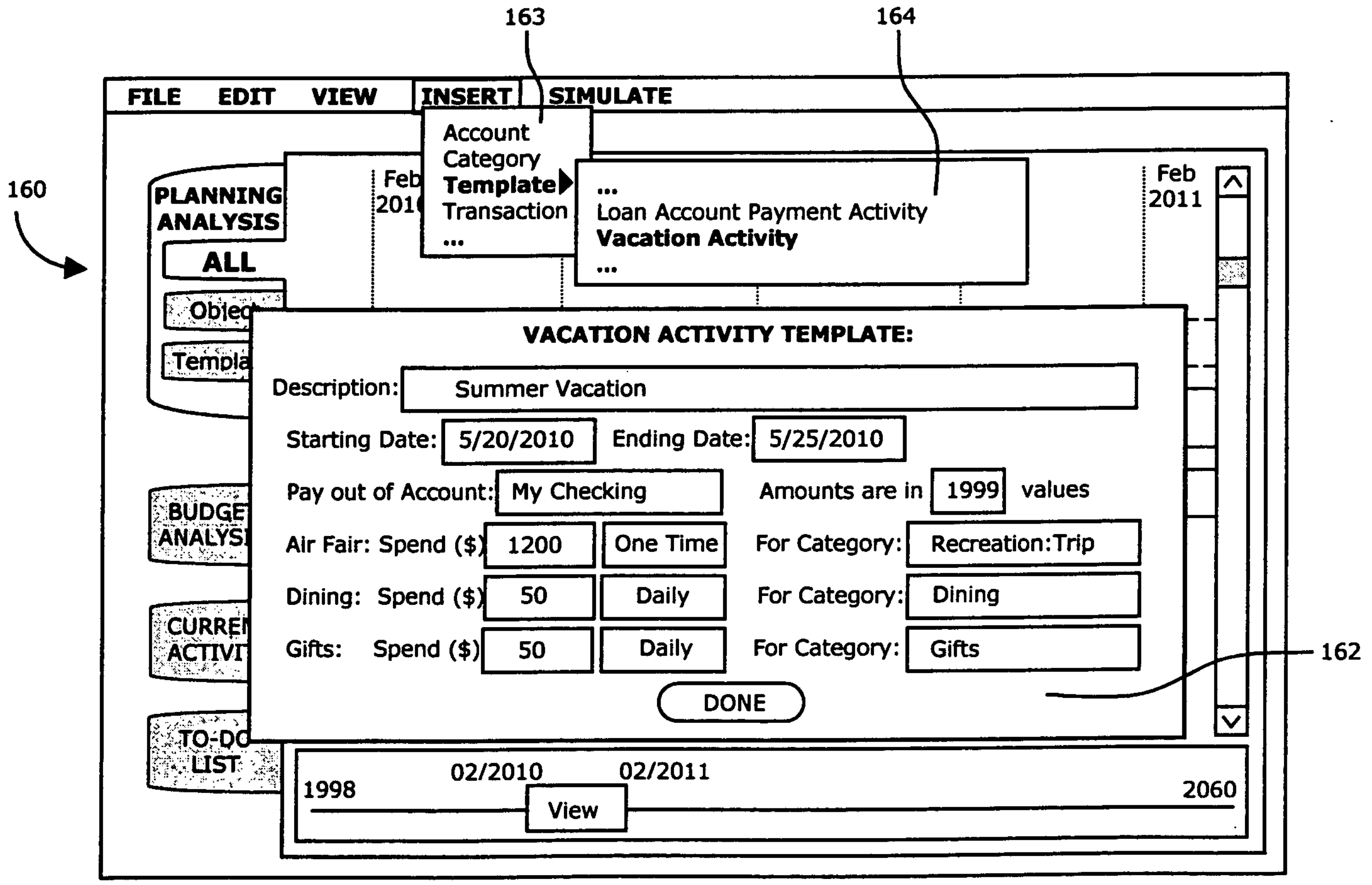


FIG. 15



VACATION ACTIVITY TEMPLATE:

Description:	Summer Vacation		
Starting Date:	5/20/2010	Ending Date:	5/25/2010
Pay out of Account:	My Checking	Amounts are in	1999 values
Air Fair: Spend ($)	1200 / One Time	For Category:	Recreation:Trip
Dining: Spend ($)	50 / Daily	For Category:	Dining
Gifts: Spend ($)	50 / Daily	For Category:	Gifts

DONE

FIG. 16

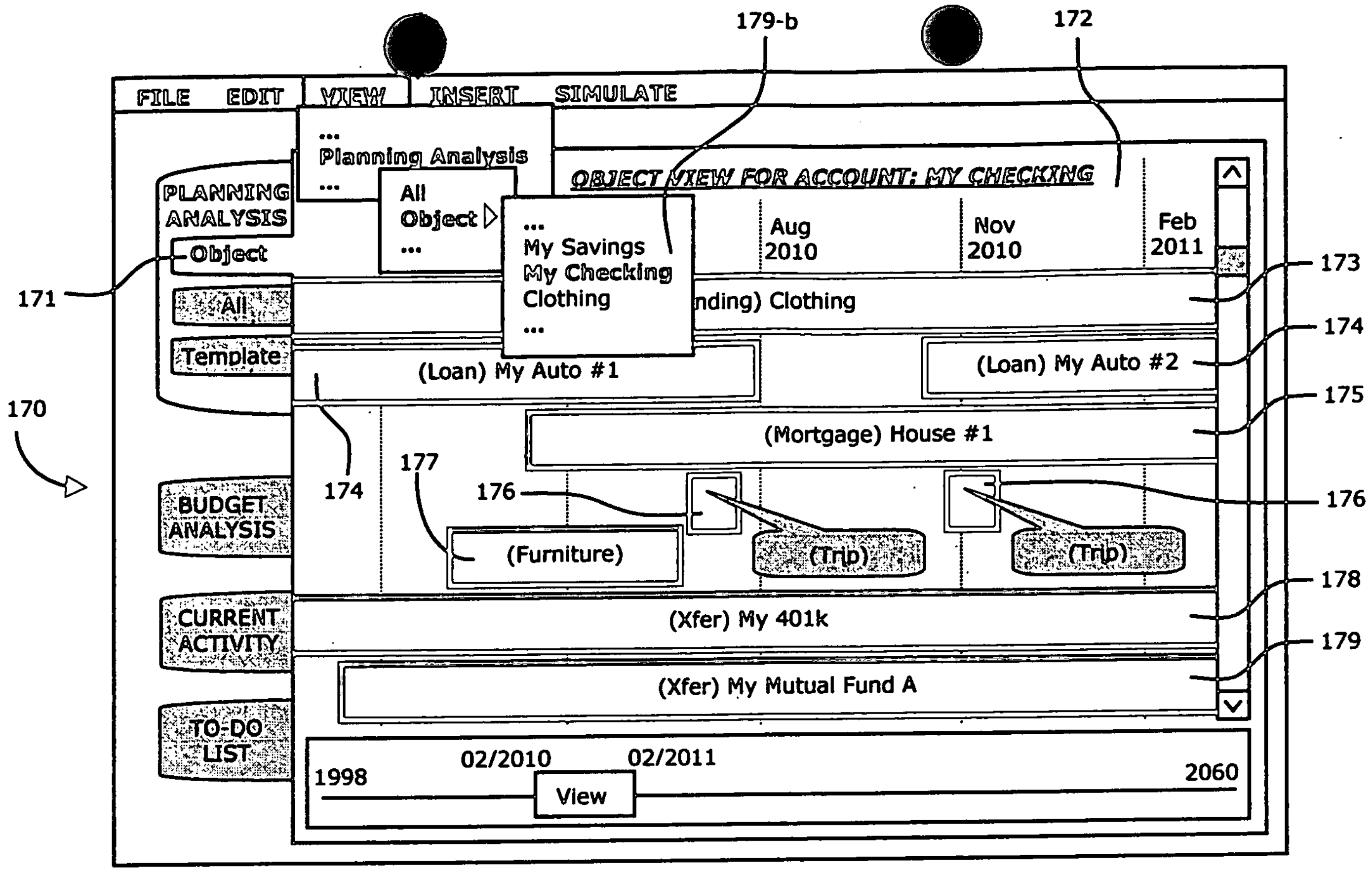


FIG. 17

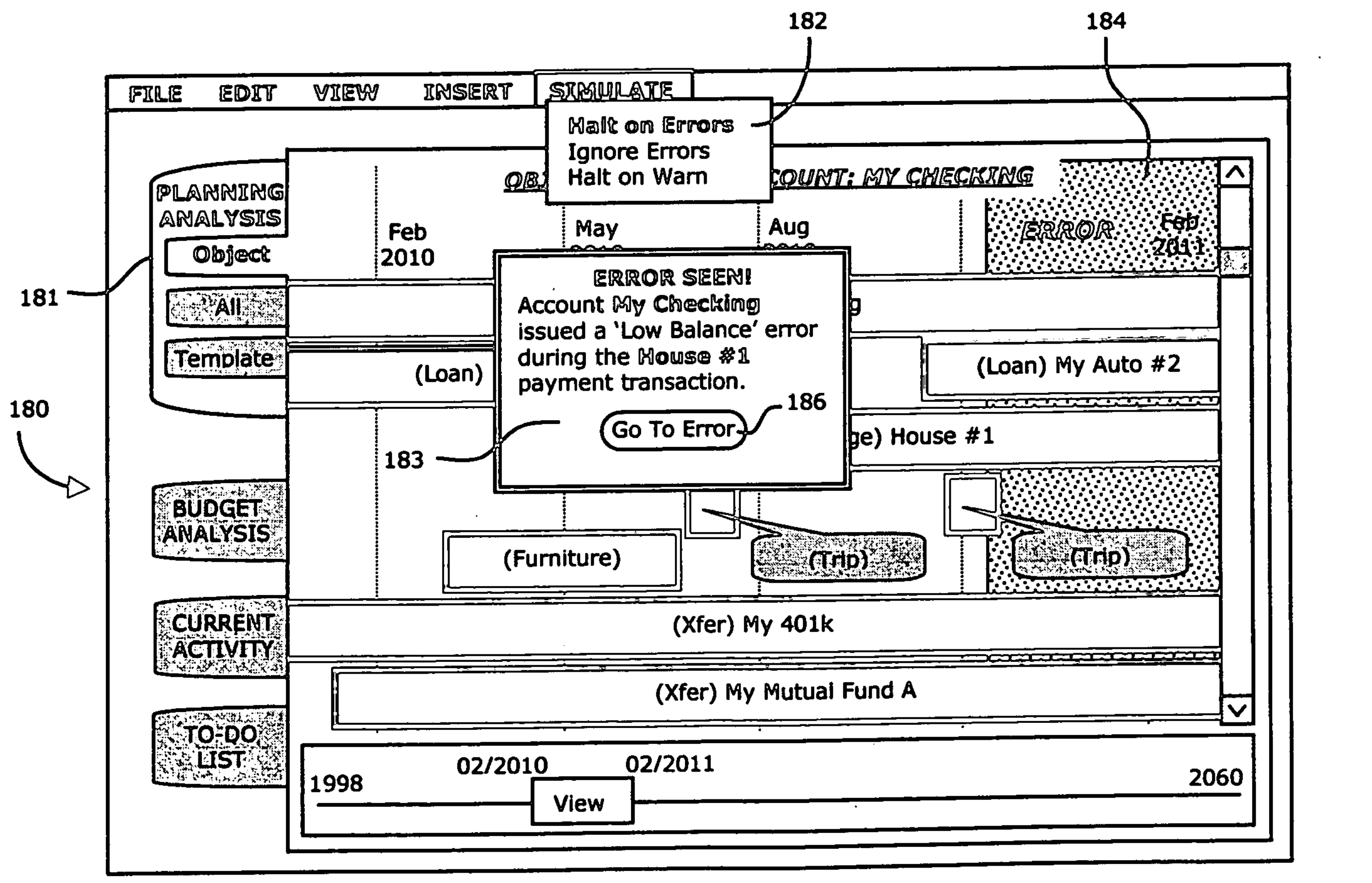


FIG. 18

BEST AVAILABLE COPY

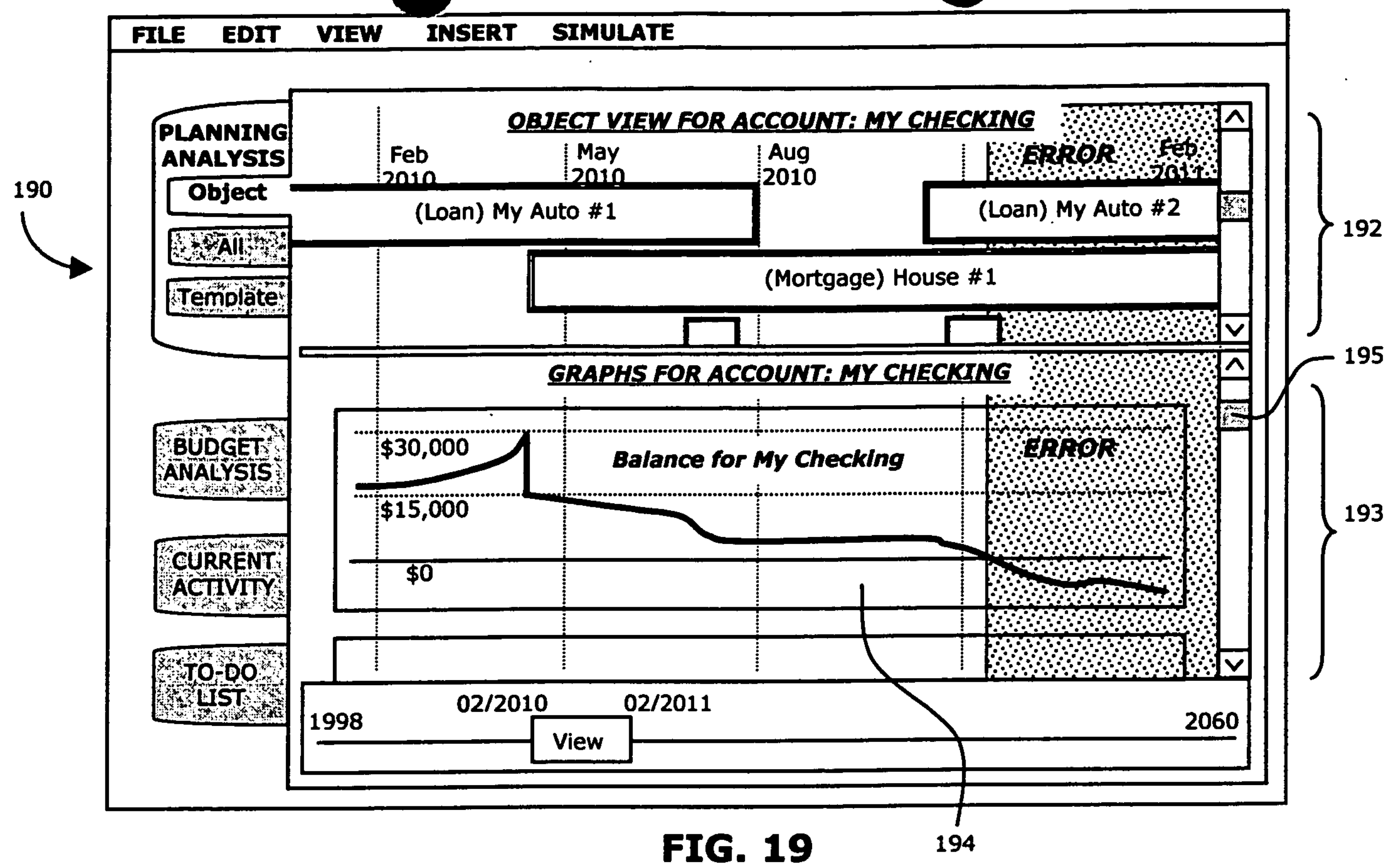

FILE
EDIT
VIEW
INSERT
SIMULATE
PLANNING ANALYSIS
Object
All
Template
BUDGET ANALYSIS
CURRENT ACTIVITY
TO-DO LIST
OBJECT VIEW FOR ACCOUNT: MY CHECKING
Feb 2010
May 2010
Aug 2010
Feb 2011
ERROR
(Loan) My Auto #1
(Loan) My Auto #2
(Mortgage) House #1
GRAPHS FOR ACCOUNT: MY CHECKING
Balance for My Checking
$30,000
$15,000
$0
ERROR
1998
02/2010
02/2011
2060
View
190
192
193
194
195

FIG. 19

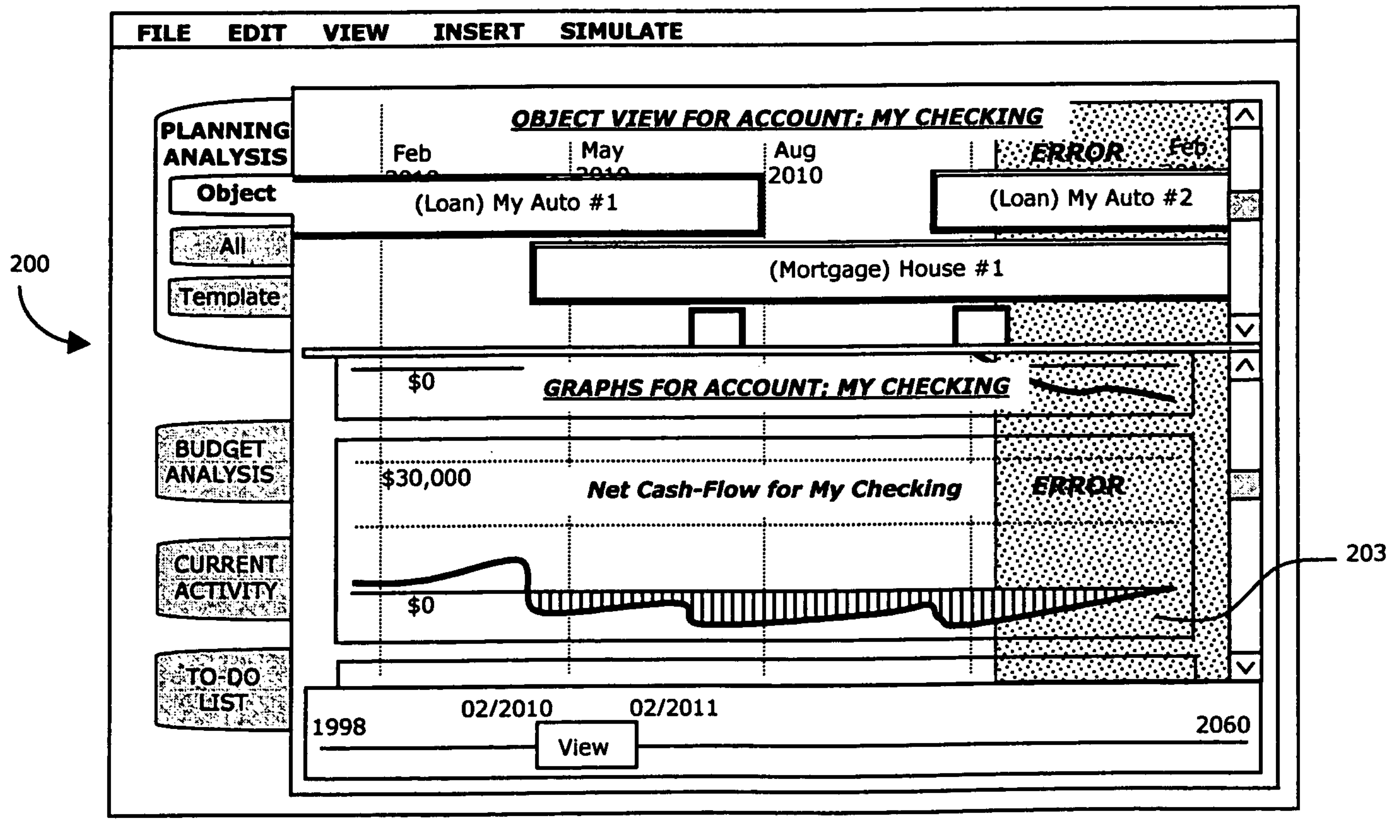
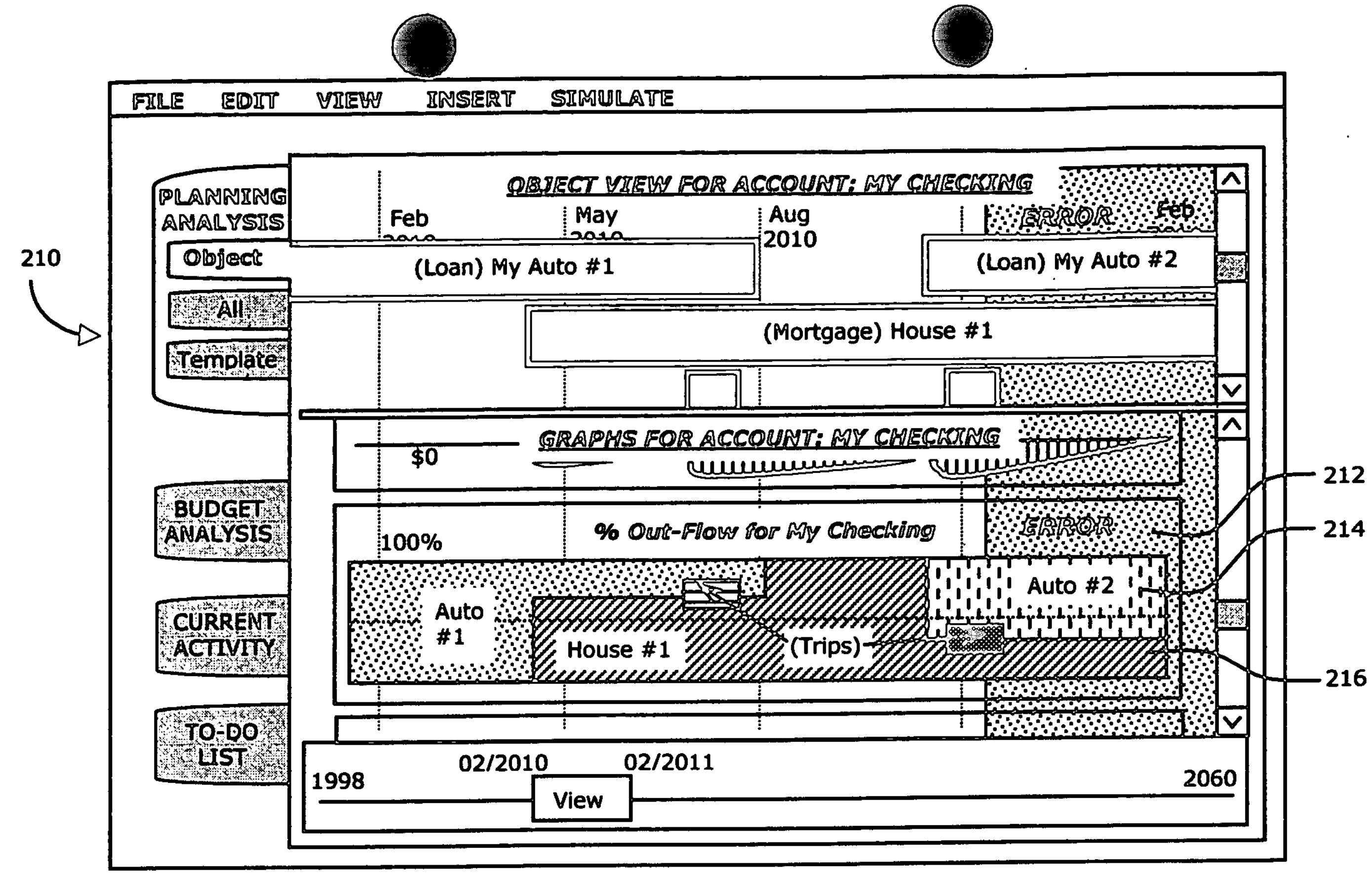
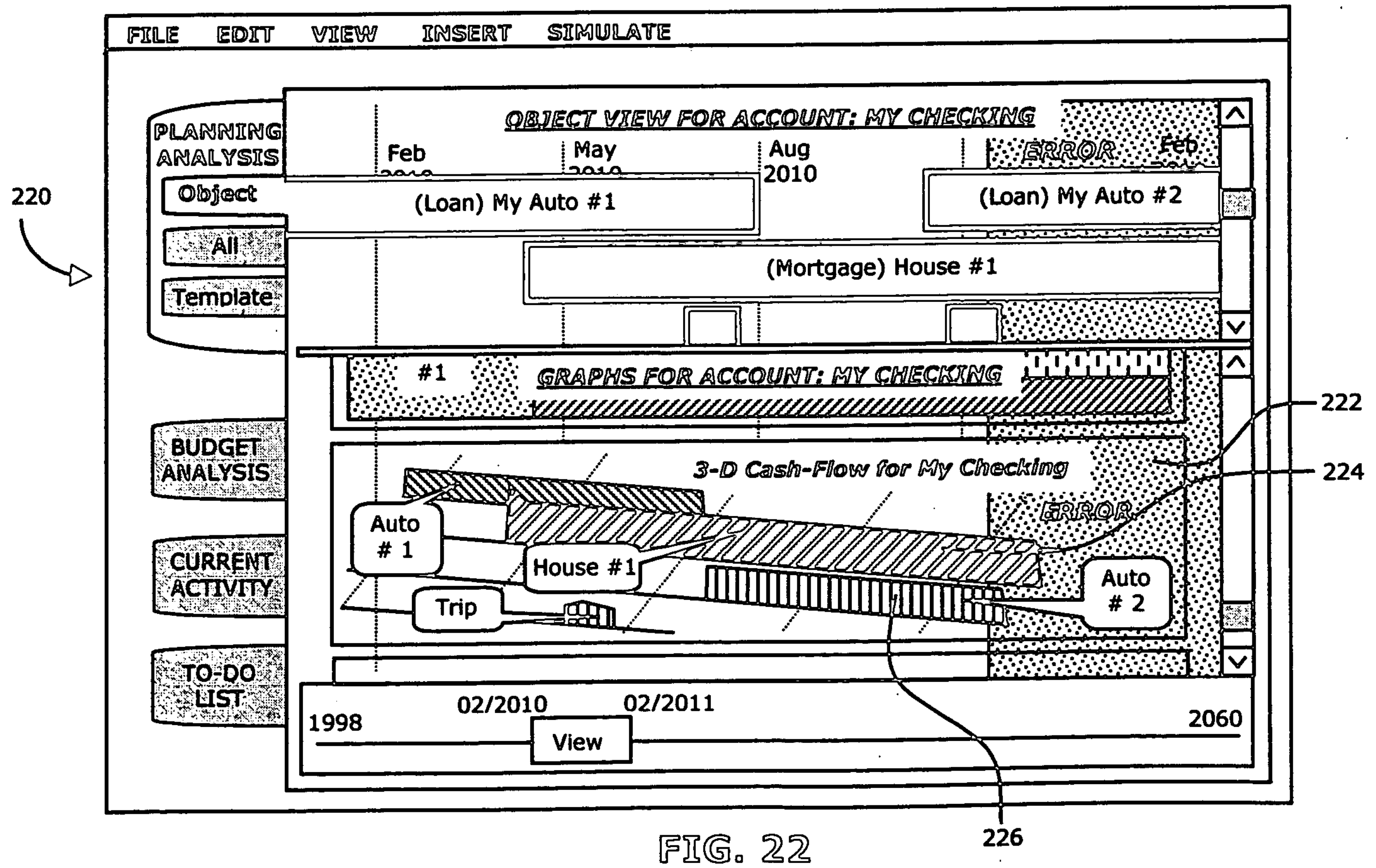

FILE
EDIT
VIEW
INSERT
SIMULATE
PLANNING ANALYSIS
Object
All
Template
BUDGET ANALYSIS
CURRENT ACTIVITY
TO-DO LIST
OBJECT VIEW FOR ACCOUNT: MY CHECKING
Feb 2010
May 2010
Aug 2010
Feb 2011
ERROR
(Loan) My Auto #1
(Loan) My Auto #2
(Mortgage) House #1
GRAPHS FOR ACCOUNT: MY CHECKING
$0
Net Cash-Flow for My Checking
$30,000
$0
ERROR
1998
02/2010
02/2011
2060
View
200
203

FIG. 20



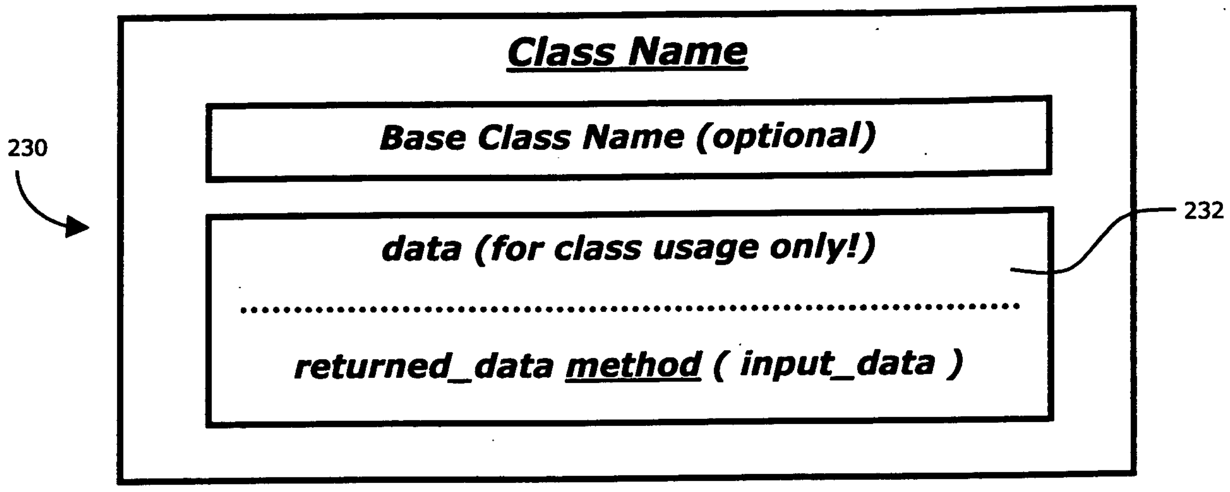
FILE
EDIT
VIEW
INSERT
SIMULATE
PLANNING ANALYSIS
Object
All
Template
BUDGET ANALYSIS
CURRENT ACTIVITY
TO-DO LIST
OBJECT VIEW FOR ACCOUNT: MY CHECKING
Feb 2010
May 2010
Aug 2010
ERROR
Feb
(Loan) My Auto #1
(Loan) My Auto #2
(Mortgage) House #1
GRAPHS FOR ACCOUNT: MY CHECKING
$0
% Out-Flow for My Checking
100%
ERROR
Auto #1
House #1
(Trips)
Auto #2
1998
02/2010
02/2011
View
2060
210
212
214
216

FIG. 21



FILE
EDIT
VIEW
INSERT
SIMULATE
PLANNING ANALYSIS
Object
All
Template
BUDGET ANALYSIS
CURRENT ACTIVITY
TO-DO LIST
OBJECT VIEW FOR ACCOUNT: MY CHECKING
Feb 2010
May 2010
Aug 2010
ERROR
Feb
(Loan) My Auto #1
(Loan) My Auto #2
(Mortgage) House #1
#1
GRAPHS FOR ACCOUNT: MY CHECKING
3-D Cash-Flow for My Checking
ERROR
Auto # 1
House #1
Trip
Auto # 2
1998
02/2010
02/2011
View
2060
220
222
224
226

FIG. 22



230
Class Name
Base Class Name (optional)
data (for class usage only!)
returned_data method (input_data)
232

FIG. 23

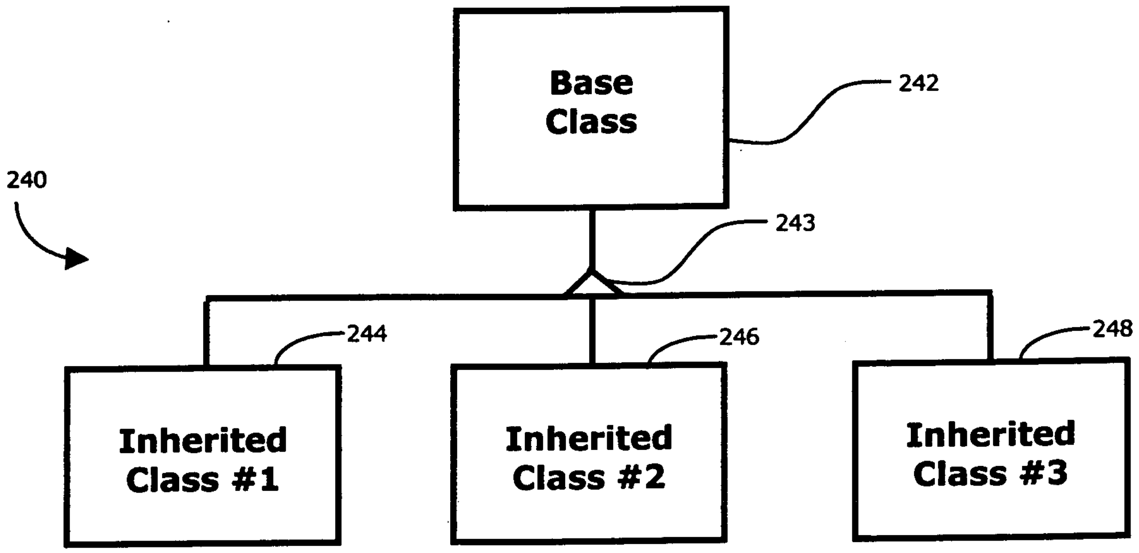


FIG. 24

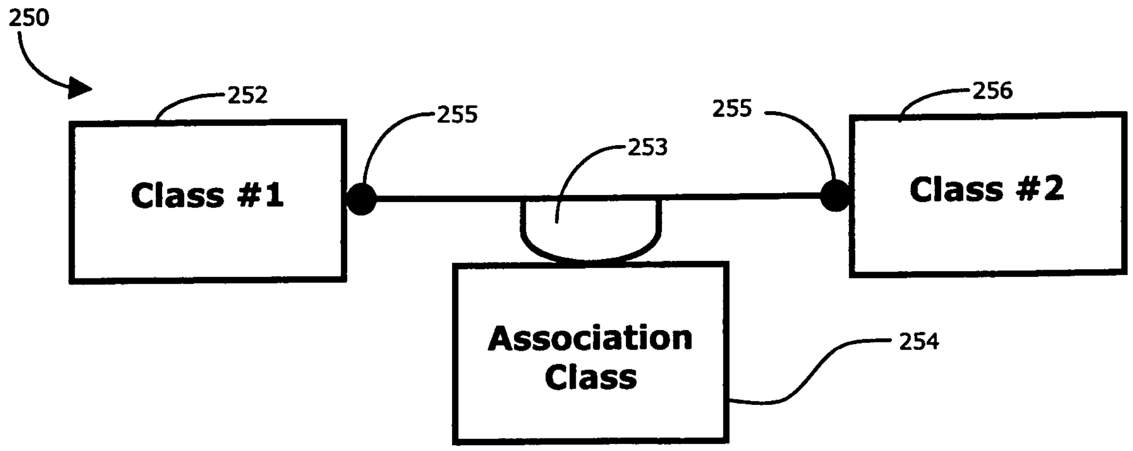


FIG. 25

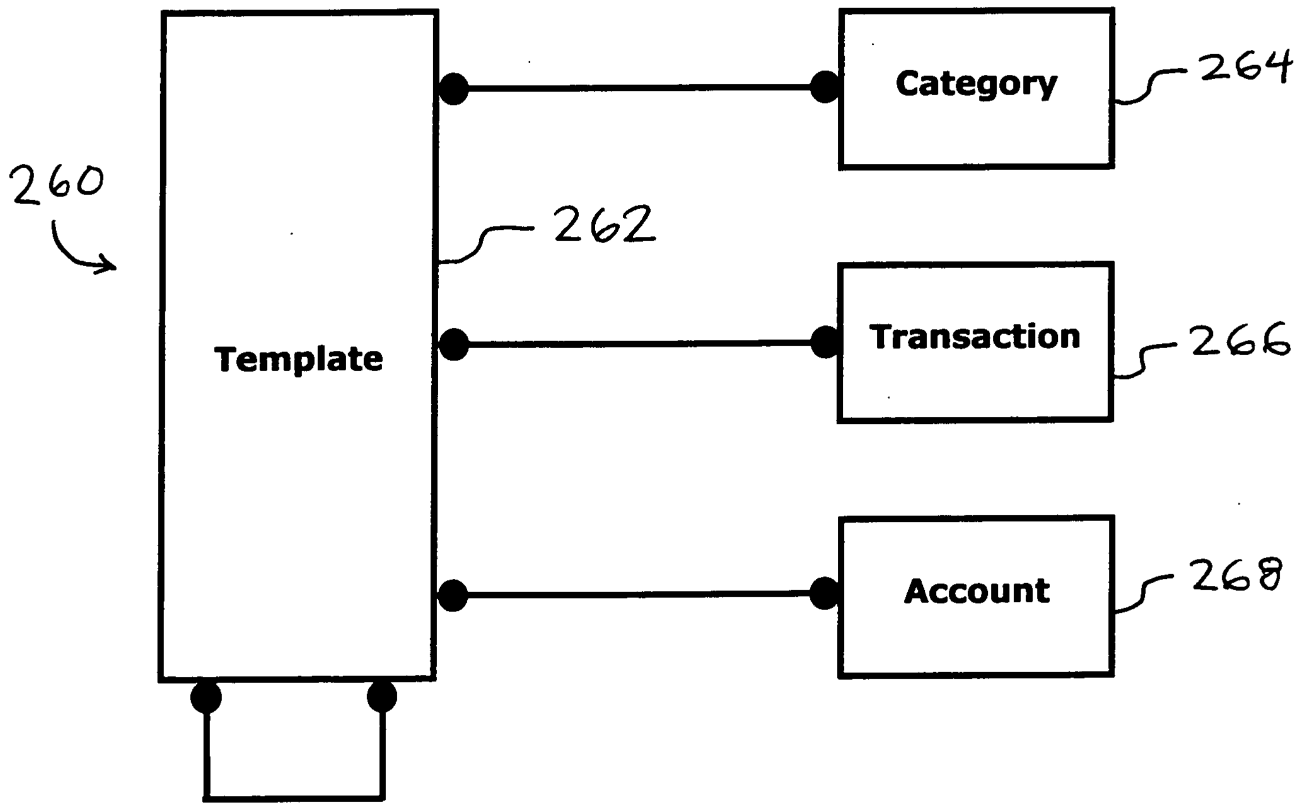


FIG. 26

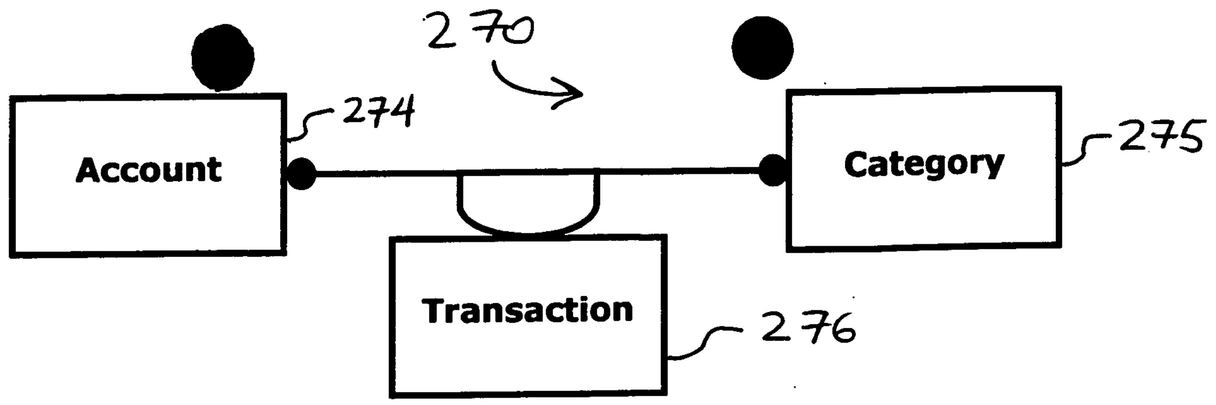


FIG. 27A

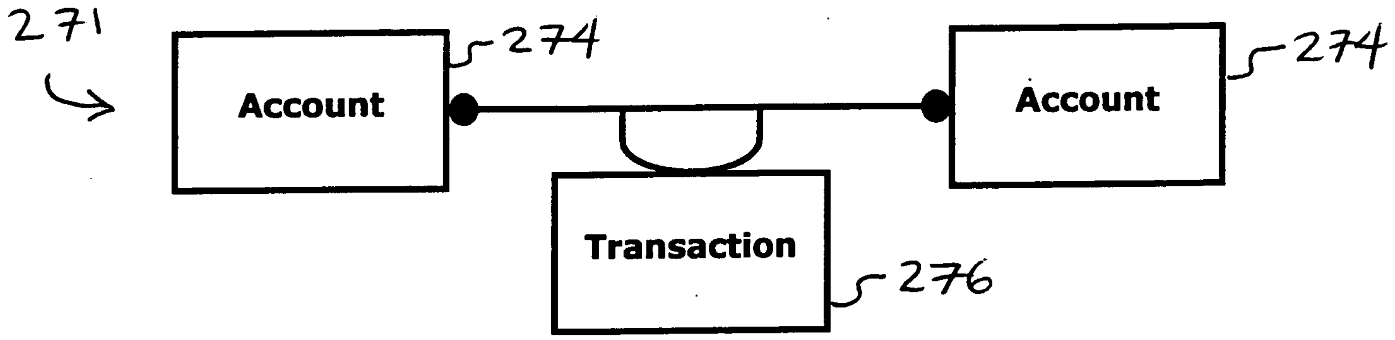


FIG. 27B

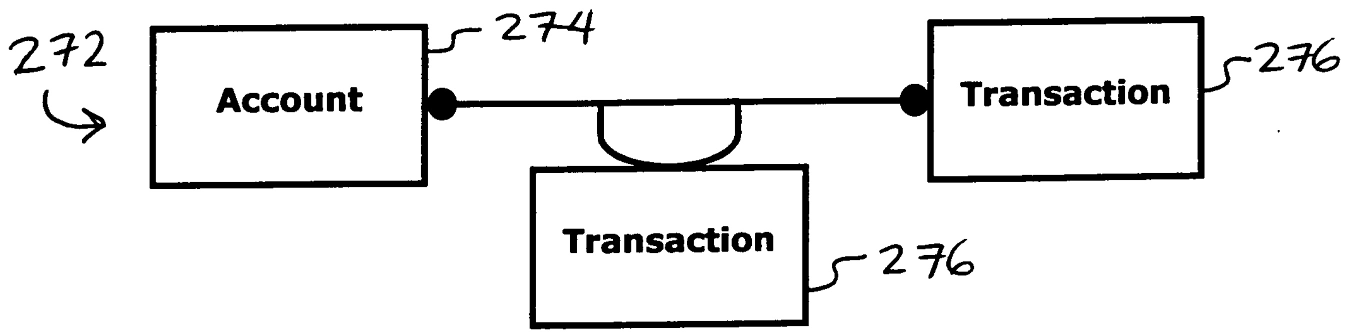


FIG. 27C

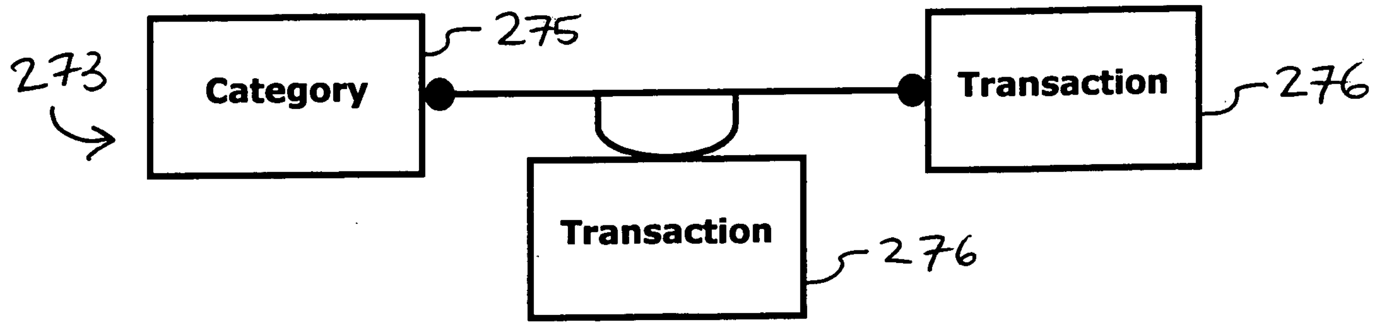


FIG. 27D

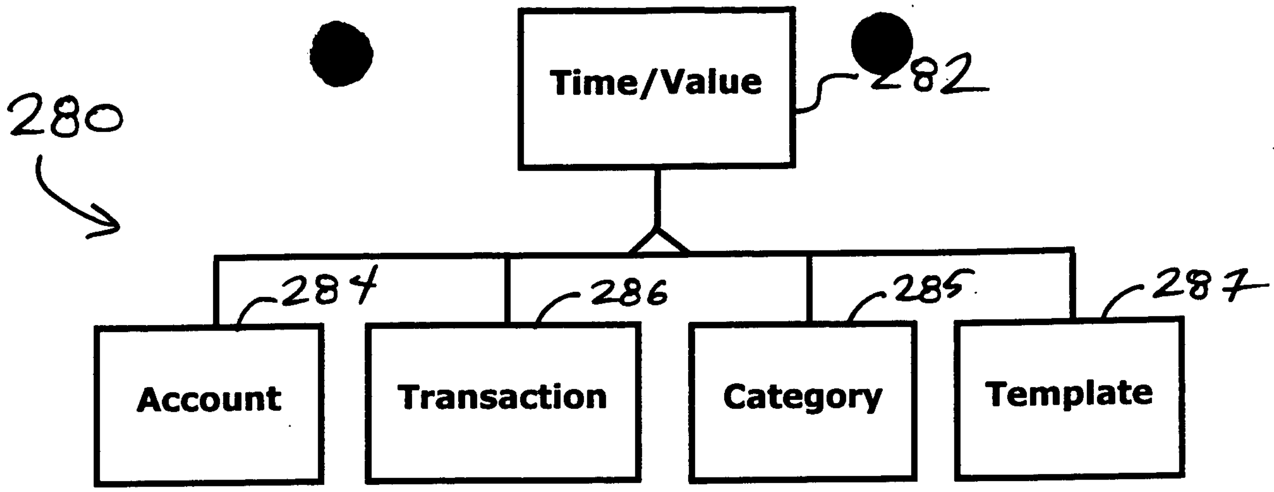
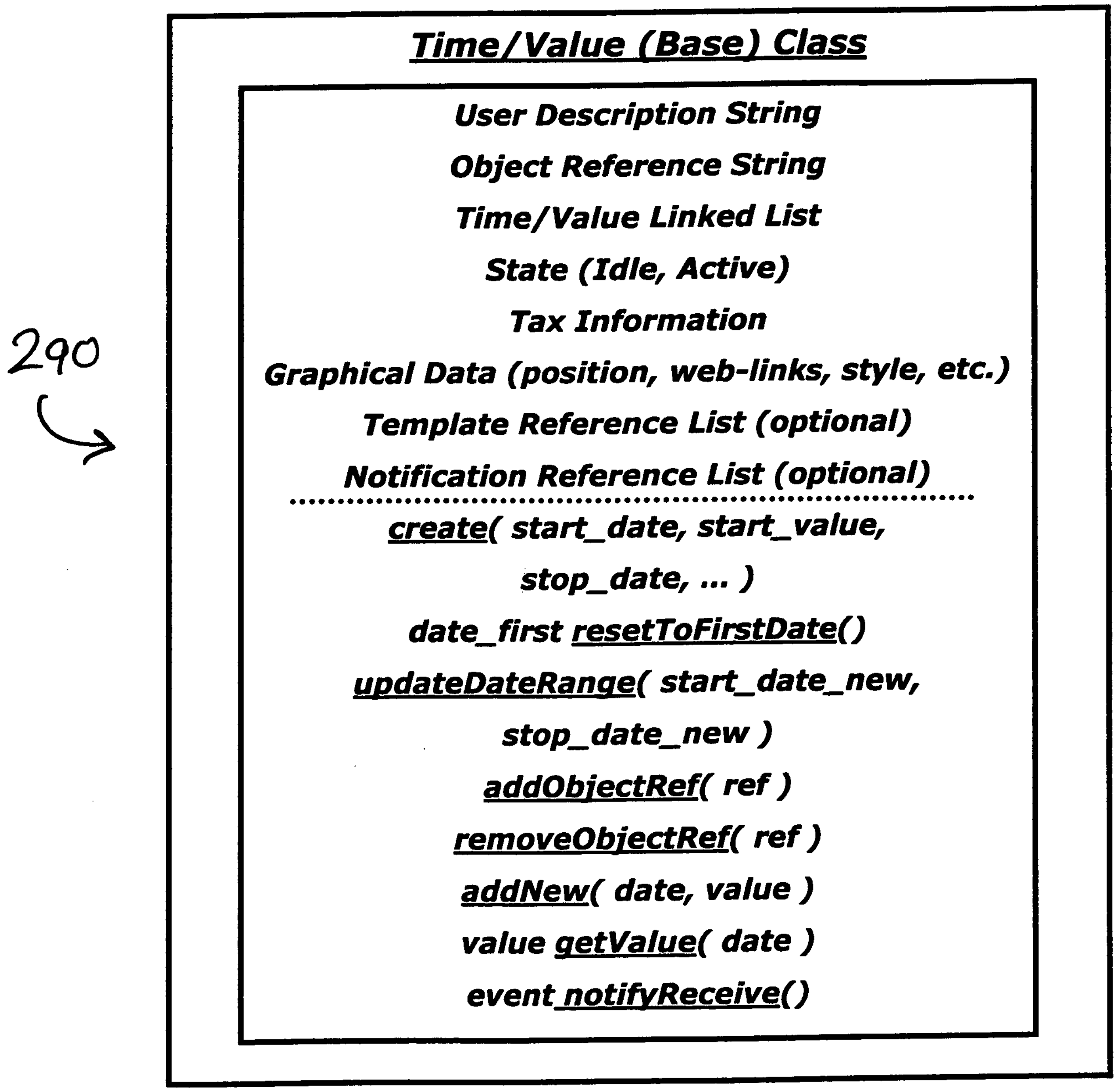


FIG. 28

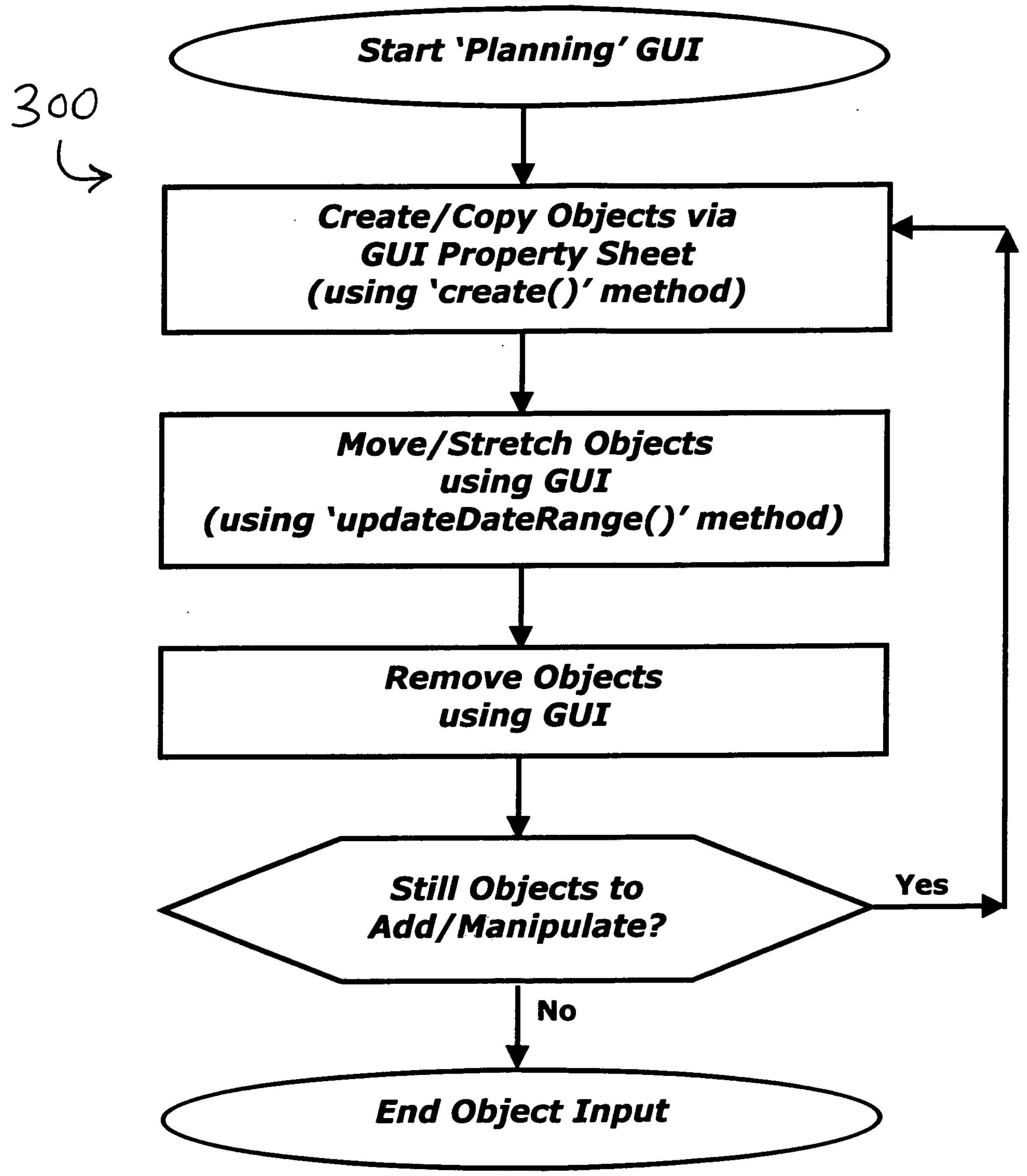


Time/Value (Base) Class

User Description String
Object Reference String
Time/Value Linked List
State (Idle, Active)
Tax Information
Graphical Data (position, web-links, style, etc.)
Template Reference List (optional)
Notification Reference List (optional)

create(start_date, start_value, stop_date, ...)
date_first resetToFirstDate()
updateDateRange(start_date_new, stop_date_new)
addObjectRef(ref)
removeObjectRef(ref)
addNew(date, value)
value getValue(date)
event notifyReceive()

FIG. 29



300
Start 'Planning' GUI
Create/Copy Objects via GUI Property Sheet (using 'create()' method)
Move/Stretch Objects using GUI (using 'updateDateRange()' method)
Remove Objects using GUI
Still Objects to Add/Manipulate?
Yes
No
End Object Input

FIG. 30

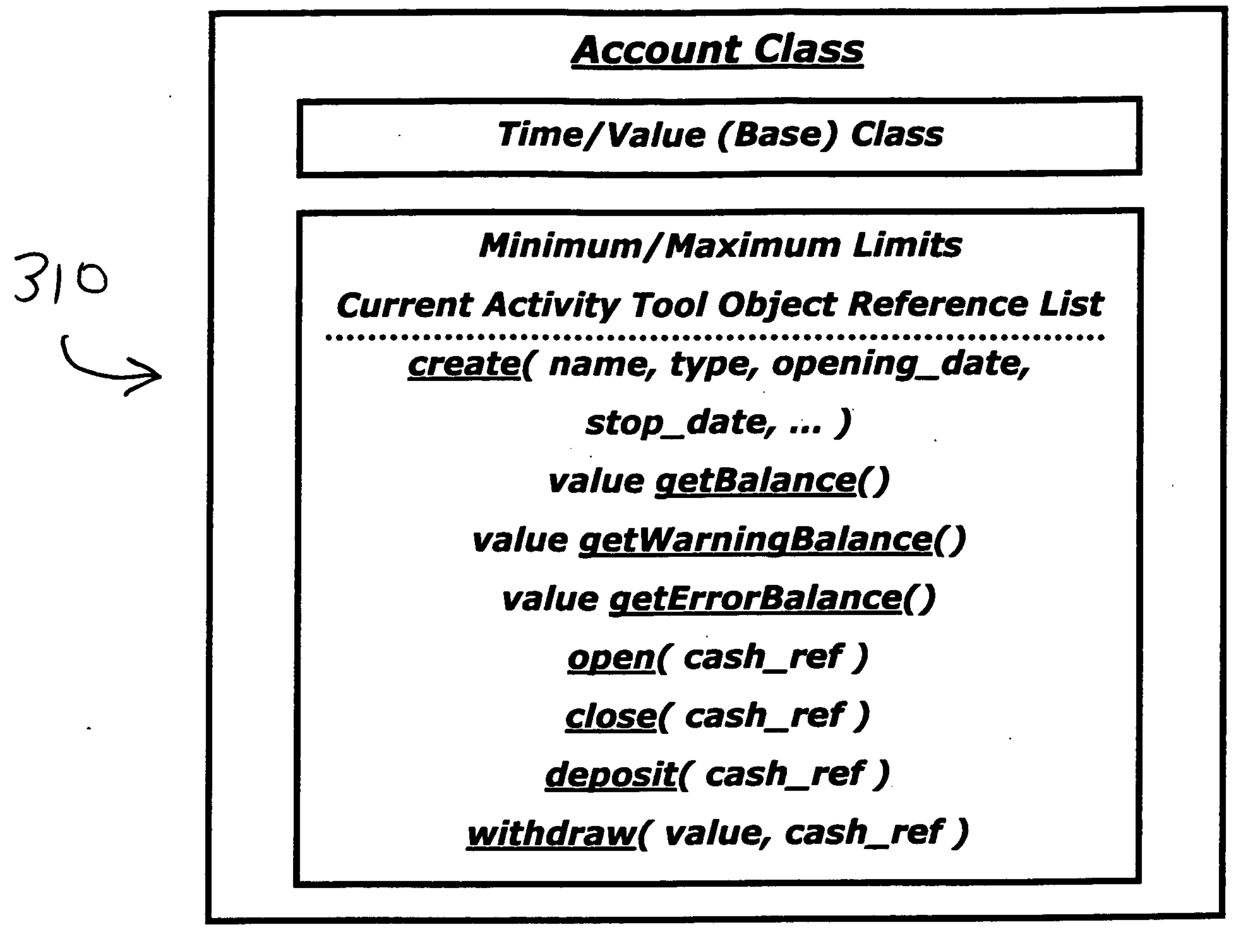

Account Class
Time/Value (Base) Class
Minimum/Maximum Limits
Current Activity Tool Object Reference List
create(name, type, opening_date, stop_date, ...)
value getBalance()
value getWarningBalance()
value getErrorBalance()
open(cash_ref)
close(cash_ref)
deposit(cash_ref)
withdraw(value, cash_ref)
310

FIG. 31

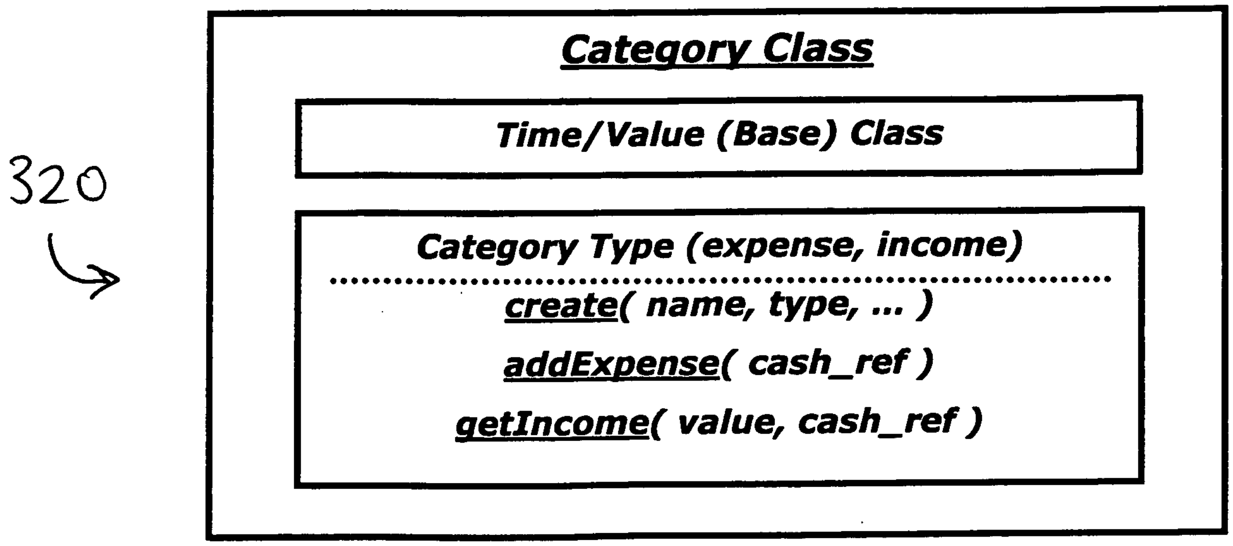

Category Class
Time/Value (Base) Class
Category Type (expense, income)
create(name, type, ...)
addExpense(cash_ref)
getIncome(value, cash_ref)
320

FIG. 32

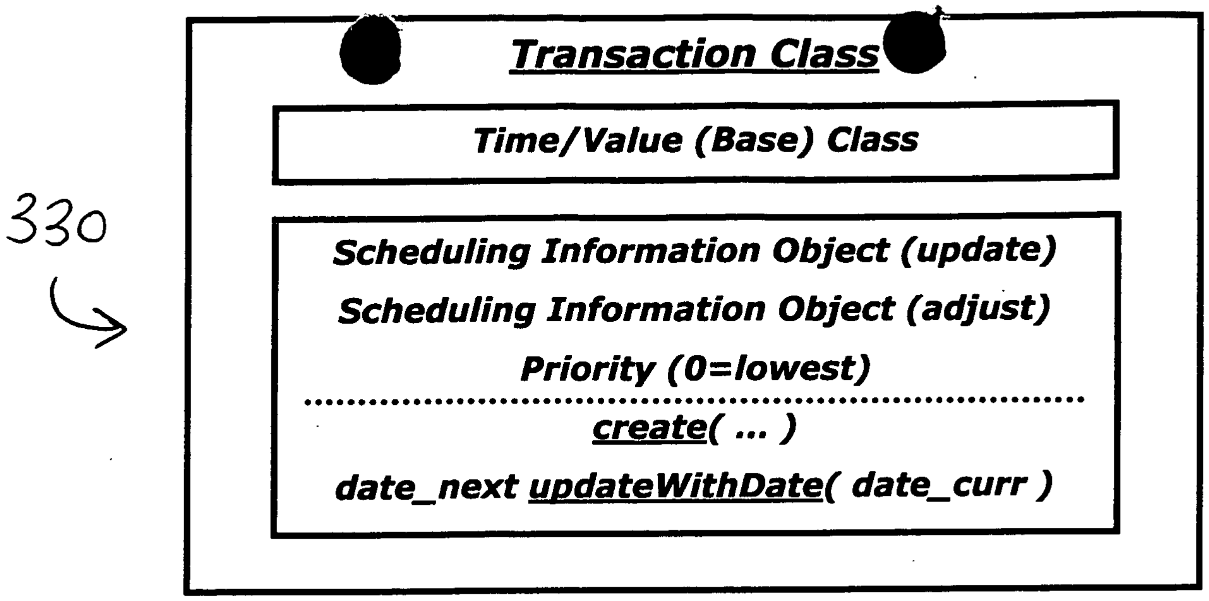

Transaction Class
Time/Value (Base) Class
330
Scheduling Information Object (update)
Scheduling Information Object (adjust)
Priority (0=lowest)
create(...)
date_next updateWithDate(date_curr)

FIG. 33

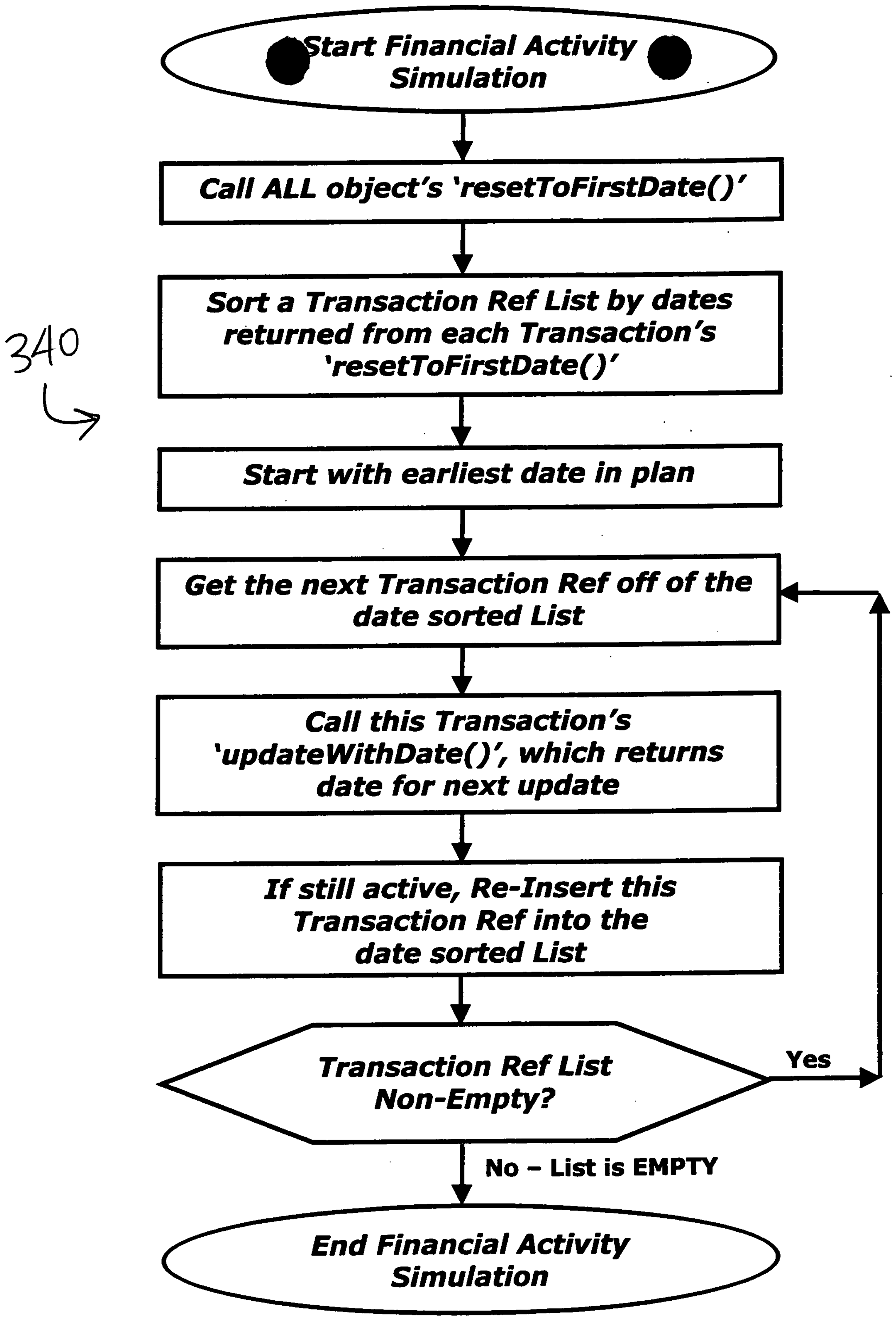
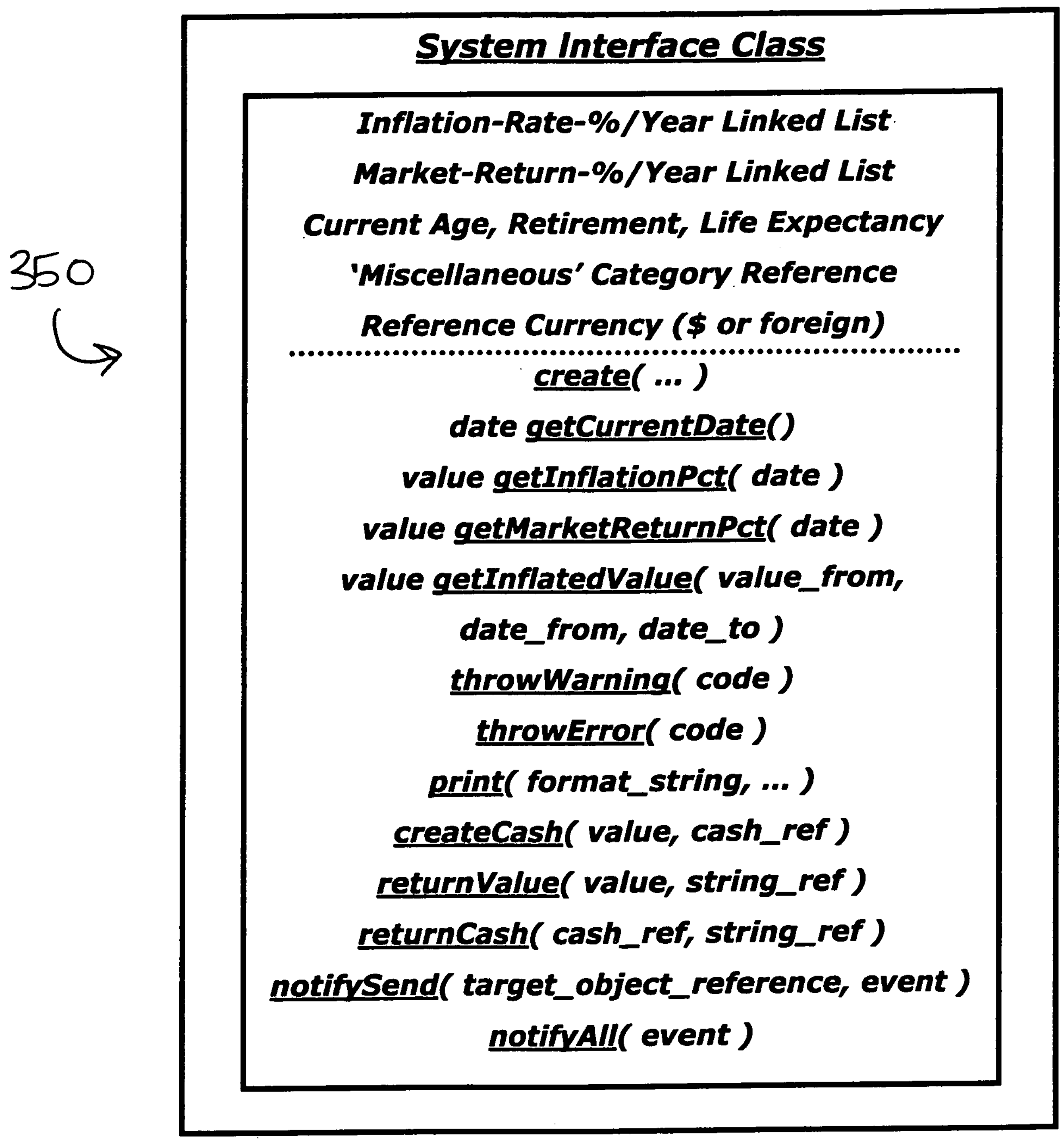

Start Financial Activity Simulation
Call ALL object's 'resetToFirstDate()'
Sort a Transaction Ref List by dates returned from each Transaction's 'resetToFirstDate()'
340
Start with earliest date in plan
Get the next Transaction Ref off of the date sorted List
Call this Transaction's 'updateWithDate()', which returns date for next update
If still active, Re-Insert this Transaction Ref into the date sorted List
Transaction Ref List Non-Empty?
Yes
No – List is EMPTY
End Financial Activity Simulation

FIG. 34



<u>***System Interface Class***</u>

Inflation-Rate-%/Year Linked List
Market-Return-%/Year Linked List
Current Age, Retirement, Life Expectancy
'Miscellaneous' Category Reference
Reference Currency ($ or foreign)

```
create( ... )
date getCurrentDate()
value getInflationPct( date )
value getMarketReturnPct( date )
value getInflatedValue( value_from,
date_from, date_to )
throwWarning( code )
throwError( code )
print( format_string, ... )
createCash( value, cash_ref )
returnValue( value, string_ref )
returnCash( cash_ref, string_ref )
notifySend( target_object_reference, event )
notifyAll( event )
```

FIG. 35

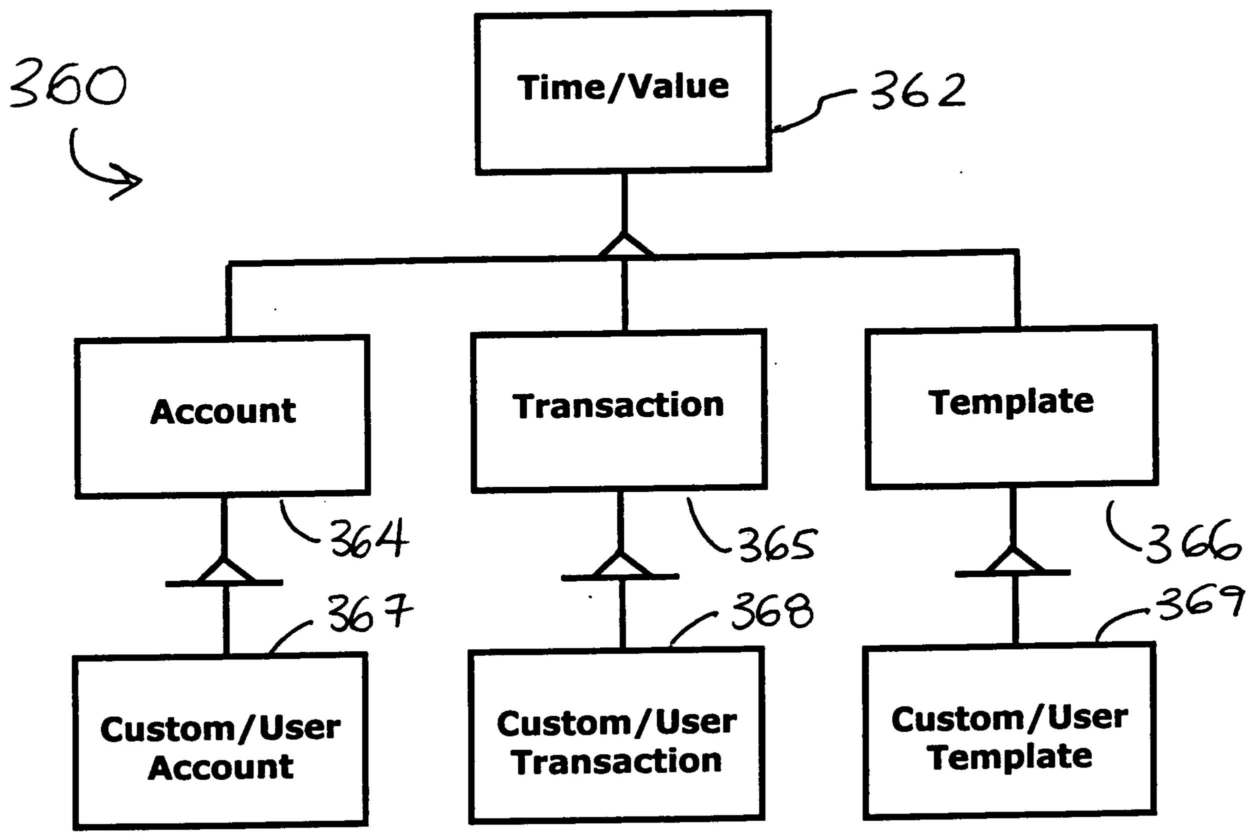

360
Time/Value
362
Account
Transaction
Template
364
365
366
367
368
369
Custom/User Account
Custom/User Transaction
Custom/User Template

FIG. 36

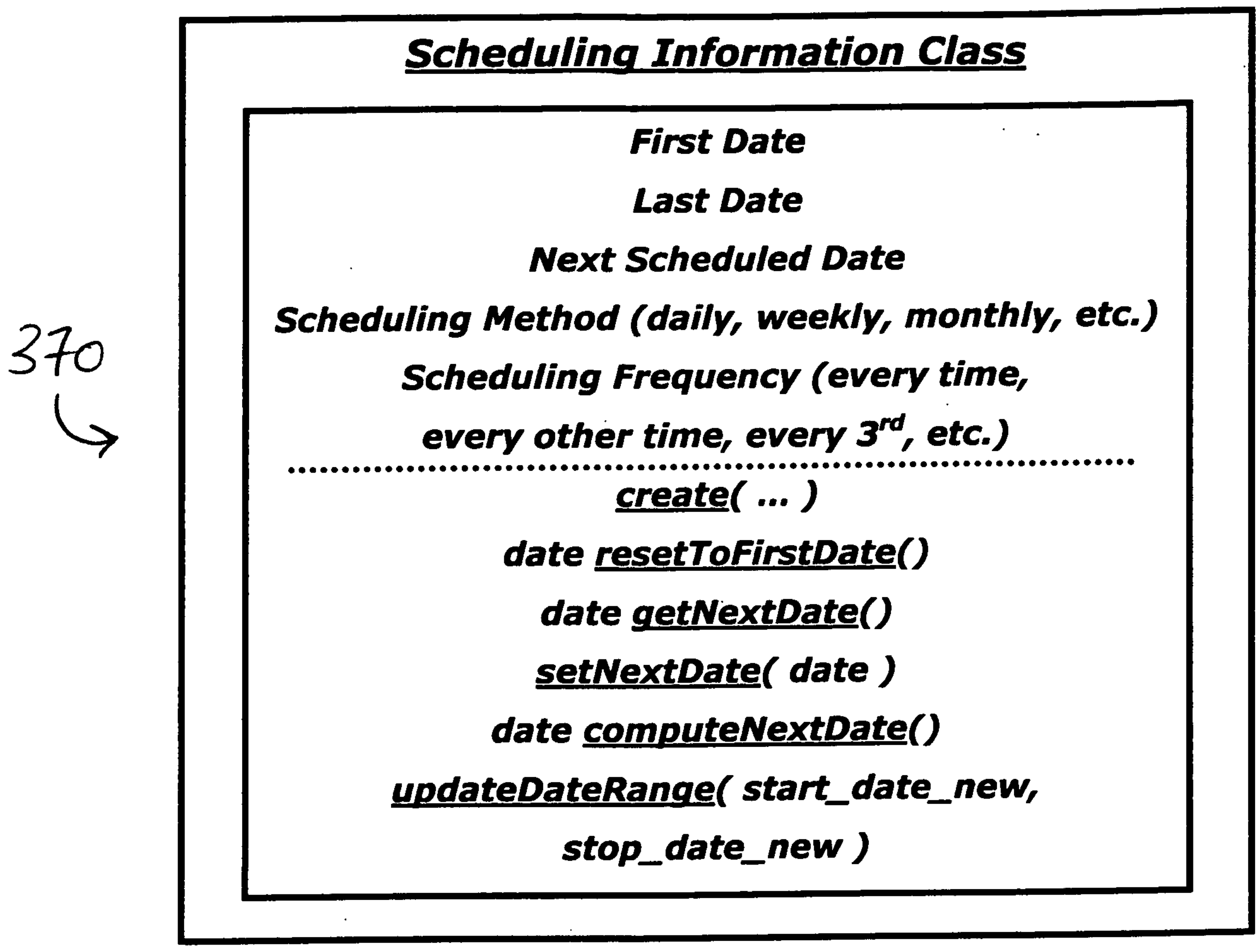

Scheduling Information Class
First Date
Last Date
Next Scheduled Date
Scheduling Method (daily, weekly, monthly, etc.)
Scheduling Frequency (every time, every other time, every 3rd, etc.)
create(...)
date resetToFirstDate()
date getNextDate()
setNextDate(date)
date computeNextDate()
updateDateRange(start_date_new, stop_date_new)
370

FIG. 37

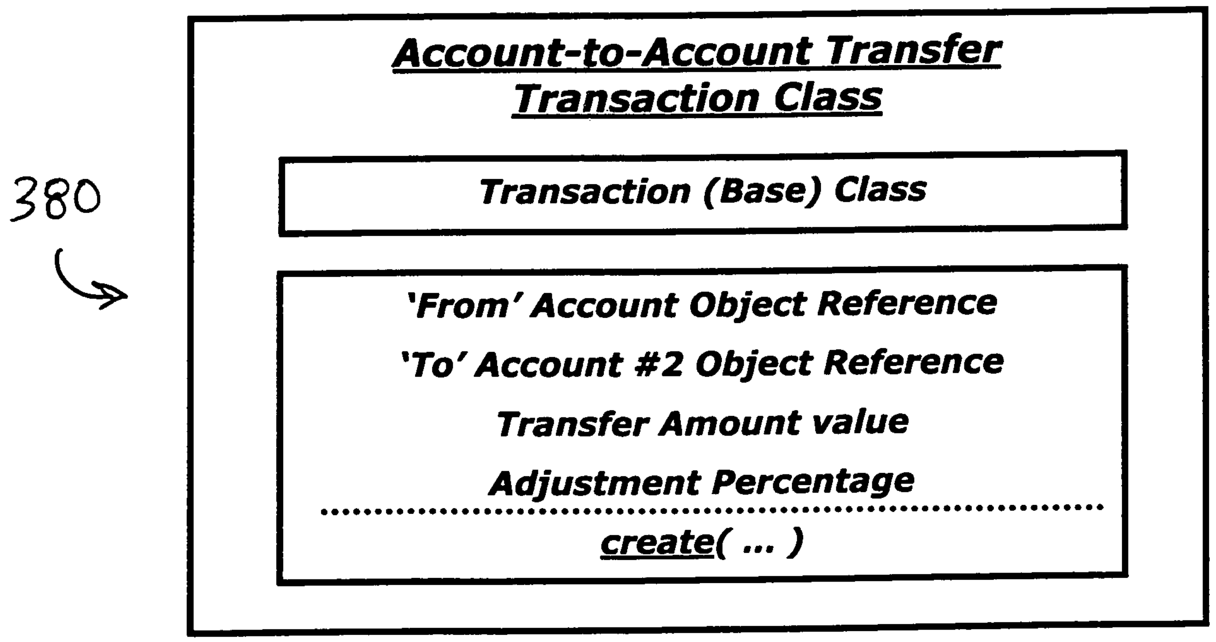
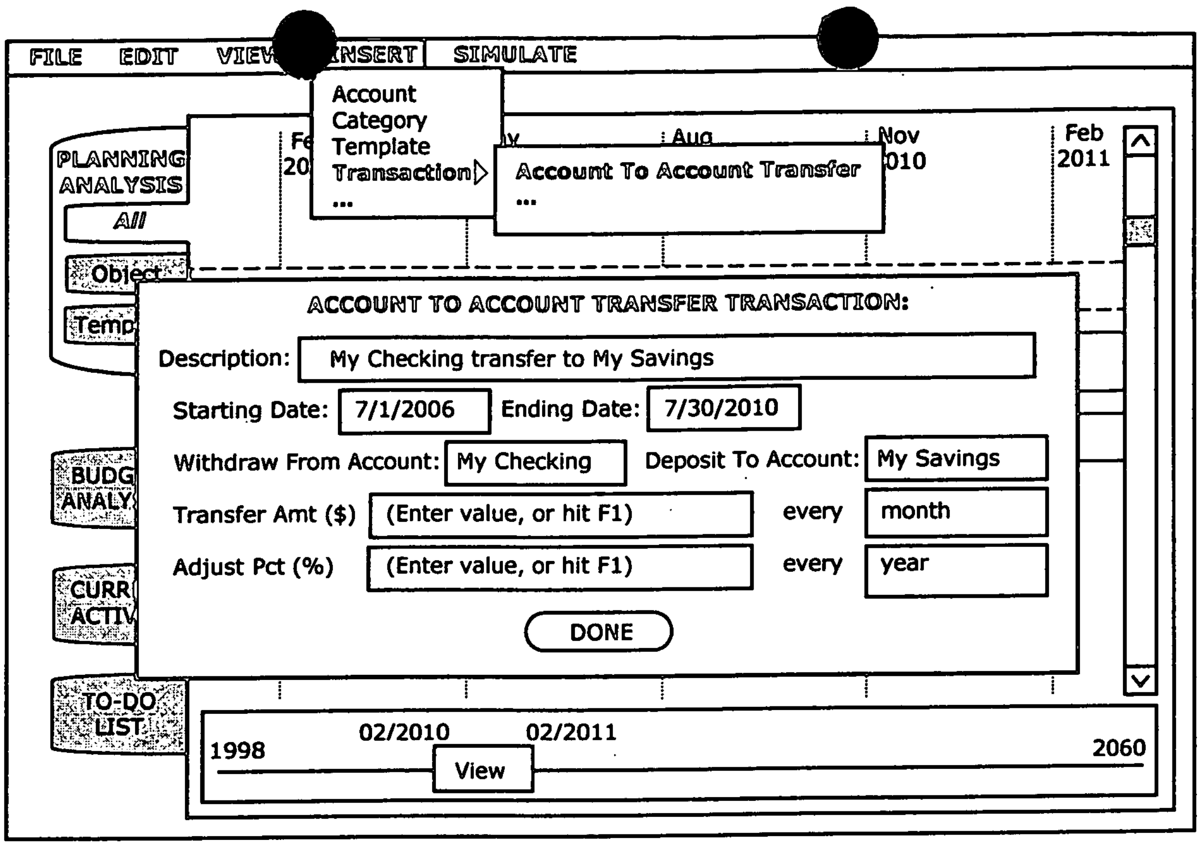

Account-to-Account Transfer Transaction Class
Transaction (Base) Class
'From' Account Object Reference
'To' Account #2 Object Reference
Transfer Amount value
Adjustment Percentage
create(...)
380

FIG. 38

390



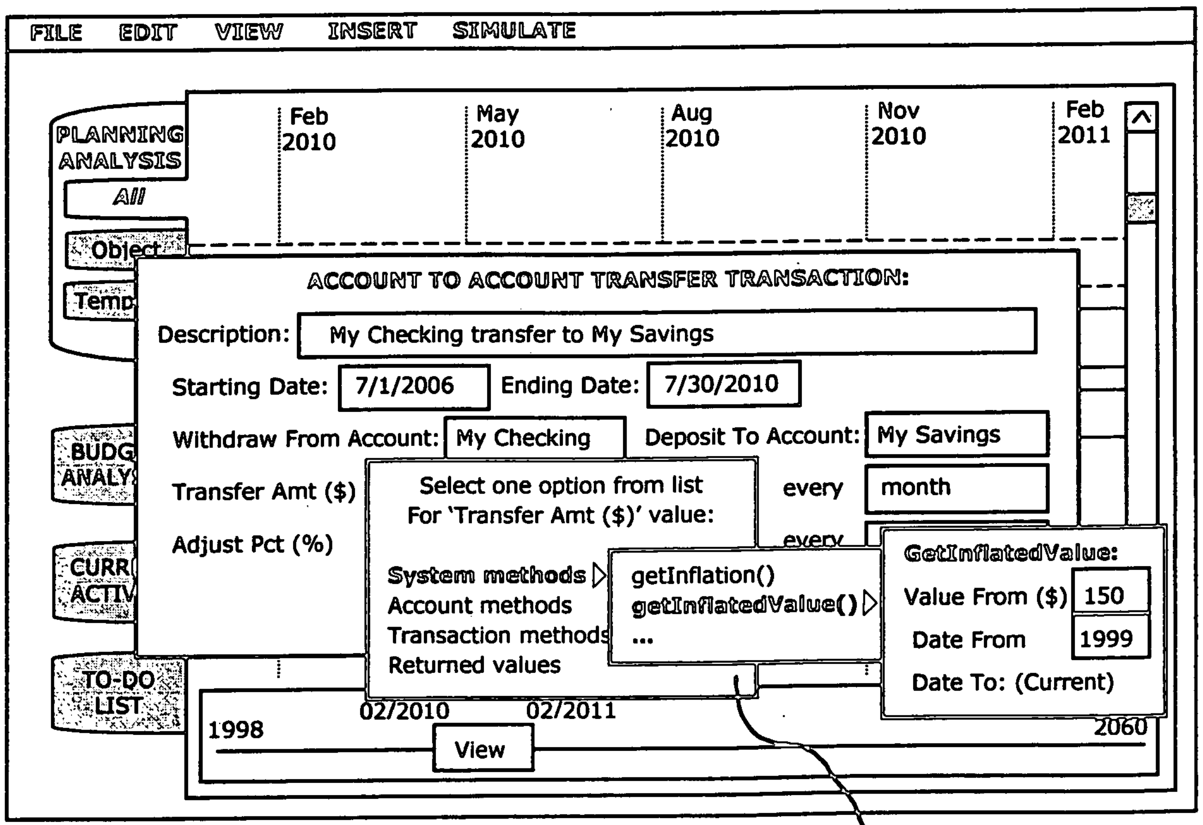
ACCOUNT TO ACCOUNT TRANSFER TRANSACTION:

Description:	My Checking transfer to My Savings		
Starting Date:	7/1/2006	Ending Date:	7/30/2010
Withdraw From Account:	My Checking	Deposit To Account:	My Savings
Transfer Amt ($)	(Enter value, or hit F1)	every	month
Adjust Pct (%)	(Enter value, or hit F1)	every	year

DONE

FIG. 39

400



ACCOUNT TO ACCOUNT TRANSFER TRANSACTION:

Description:	My Checking transfer to My Savings		
Starting Date:	7/1/2006	Ending Date:	7/30/2010
Withdraw From Account:	My Checking	Deposit To Account:	My Savings
Transfer Amt ($)		every	month
Adjust Pct (%)		every	

GetInflatedValue:

Value From ($) 150

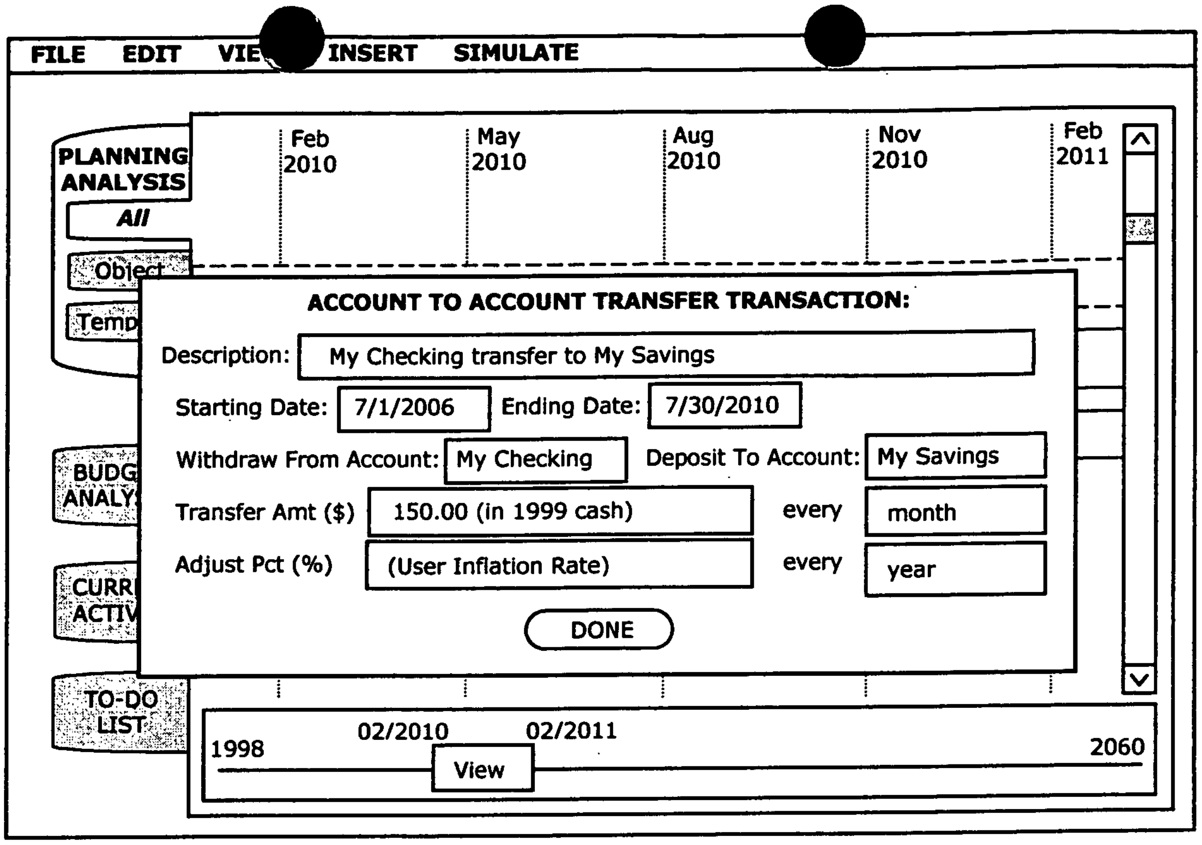
Date From 1999

Date To: (Current)

402

FIG. 40

410



ACCOUNT TO ACCOUNT TRANSFER TRANSACTION:

Description: My Checking transfer to My Savings

Starting Date: 7/1/2006 Ending Date: 7/30/2010

Withdraw From Account: My Checking Deposit To Account: My Savings

Transfer Amt ($) 150.00 (in 1999 cash) every month

Adjust Pct (%) (User Inflation Rate) every year

DONE

FIG. 41

420

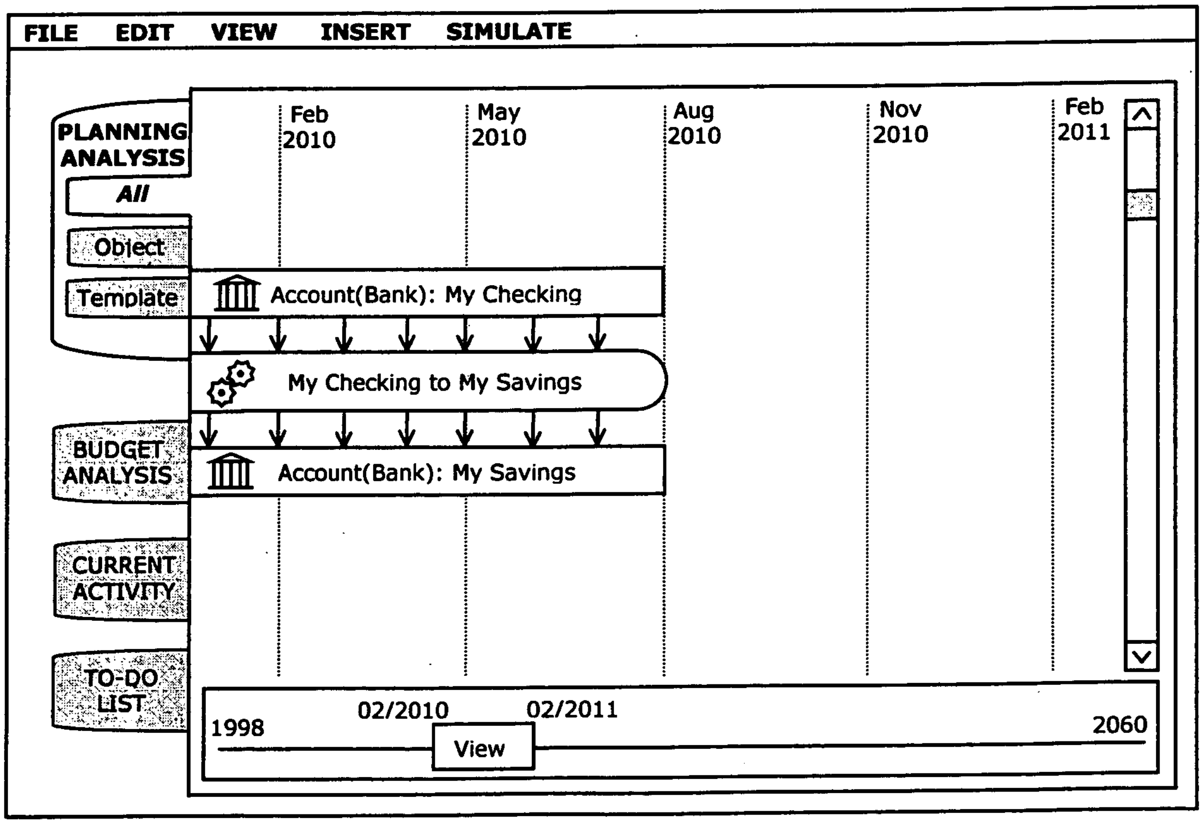


FIG. 42

Transaction Class

430

```
class CTrans : public CTimeValue          // A pure-virtual/abstract base class!
{
public:
   CTrans(
       const char     *name,              // Transaction's reference-name
       ...                                // More input parameters (not shown)
         );
   virtual ~CTrans();

   virtual CDate updateWithDate( CDate date_curr ) = 0; // PURE VIRTUAL!
                                                    // ...Must inherit this class!

   Virtual CDate resetToFirstDate();                // Inherited from Time/Value
   Virtual void updateDateRange( CDate date_start, CDate date_stop );
                                                    // Inherited from Time/Value
protected:
   CScheduler        m_schUpdate;         // Schedules next update date
   CScheduler        m_schAdjust;         // Schedules next adjust date
   priority_t        m_priority;          // Priority (0=lowest)

}; // END of 'CTrans' class
```

FIG. 43

Account-to-Account Transfer Transaction Class

440

```
class CTrans_acctToAcct : public CTrans
{
public:
   CTrans_acctToAcct(
       const char     *name,              // Transaction's reference-name
       ...                                // More input parameters (not shown)
                      );
   virtual ~CTrans_acctToAcct();

   virtual CDate updateWithDate( CDate date_curr );

   Virtual CDate resetToFirstDate();
   Virtual void updateDateRange( CDate date_start, CDate date_stop );

protected:
   CAccount          *m_acctFrom;         // Xfer 'From' this accnt
   CAccount          *m_acctTo;           // Xfer 'To' this accnt
   value_t           m_moneyToXfer;       // Money to transfer at each update
   value_t           m_adjustPct;         // % to adjust xfer amount

}; // END of 'CTrans_acctToAcct' class
```

FIG. 44

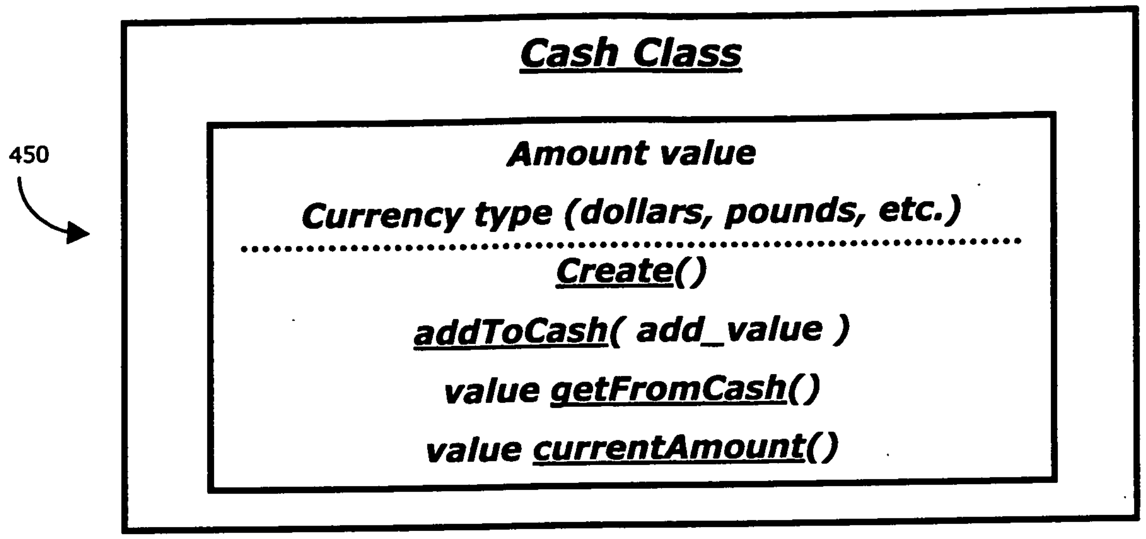

Cash Class
450
Amount value
Currency type (dollars, pounds, etc.)
Create()
addToCash(add_value)
value getFromCash()
value currentAmount()

FIG. 45

Account-to-Account Transfer Transaction Class Method Example

```
CDate CTrans_accntToAccnt::updateWithDate( CDate date_curr )
{
   Cdate date_test = SYSINTF.getCurrentDate(); // Not used here, just for demo
   if ( date_test == date_curr )
   {
        SYSINTF.print( "Just a test...dates are equal!" );
   }

   // Check to see if the simulated current date does NOT match our expected
   //    current date, leaving if it doesn't (an invalid condition)
   if ( date_curr != m_schUpdate.getNextDate() )
   {
        SYSINTF.throwError( ERR_UNEXPECTED_DATE );
        return( date_curr );  // Terminates simulation for this transaction!
   }
```

460

```
   // Adjust parameters if the current simulation date matches or exceeds
   //    our next adjustment date
   if ( date_curr >= m_schAdjust.getNextDate() )
        m_moneyToXfer *= 1.0 + m_adjustPct / 100.0;
        m_schAdjust.computeNextDate();  // Set the next adjustment date
   }

   // CREATE a 'cash' data type (sets simulated cash amount to ZERO)
   CCash cash_xfer;

   // WITHDRAW cash FROM account (makes simulate cash a positive amount)
   m_acctFrom->withdraw( m_moneyToXfer, cash_xfer );

   // DEPOSIT cash TO account (makes simulated cash zero again, after transfer)
   m_acctTo->deposit( cash_xfer );

   // LOG this transfer amount to the Time/Value (base) class
   addNew( date_curr, m_moneyToXfer );

   // Return the date that we wish the Cash-Flow Simulator to call us with again
   return( m_schUpdate.computeNextDate() );

   // NOTE: When this method call returns, 'cash_xfer' will be AUTOMATICALLY
   //    destroyed, which calls the 'cash' class' destructor method call.  A NON-ZERO
   //    simulated cash amount in 'cash_xfer' would cause a system warning!

} // END of 'CTrans_accntToAccnt::updatePerDate()'
```

FIG. 46